om the world who came and said we're gonna work on this and we're gonna help make this real uh it it's interesting and we're gonna I'm sure just cover this but uh I helped to found Varys I I was walking along the street after I developed a machine learning language model for a large company to uh recognize emails so it would read emails and it would combine it with other information and it would figure out what the email was about general you know classification of subject and things like that who might have written it other things about it and it would do all that in about 250 microseconds that was you know a little over five years ago and once I had built that I realized that uh you know with all that was happening we're headed for this world with it's gonna be really hard to figure out what's true and what's not true that was years ago when that was clear and the idea was how do you create a system and this is well not about it was not about a get rich quick or oh we can make a coin or any of that it was how do you create a system that could help you get training data from the world but still know that the data is coming from provable sources or other sources that aren't provable and and really be able to sort all of this out at scale at scale of the world so my background I was um you know I started in uh Building Systems back in the early 80s um I I built multitasking uh environments that beat windows and info World reviews and um and eventually uh found my way you know I had started some companies um built some bioses and low level operating system things and combination of Assembly Language or C or you know and and I had released some uh open software and so I found my way to Microsoft ended up being the lead for Dev lead for kernel for Windows 95 and uh started the Java platform at Microsoft by starting to write their their version of the Java VM along with some other people um and I started the.net platform uh I was one of two recognized Founders I started the platform side and another uh individual was working on the tools side at the time and you know it's always been this this uh goal of mine to make computers work for people not the other way around so when I realized that I could process an email in 250 microseconds and I could figure out all this and it wasn't you know it wasn't me it was learning how to do this um it was kind of a an alarm Bell for me we need a system like this so that's when I got some people together we decided that this was going to be a fair launch we could not make a company out of this being part of something that was you know designed to scale to the world and and be in open rent-free system that people could take and use and not have to pay anyone to use um you know that was the plan that's what we made and and we needed privacy I know I know that this series that you've got is uh is focused on privacy and that was one of the critical aspects because if we're going to have people using machine learning at scale and we're going to have you know transactions Financial transactions open so much is going to be able to be understood about you as an individual I mean I was the technical fellow for the Microsoft advertising platform we used to know more about most people back then than they might have known about themselves and the fact is that they didn't really have a clue about how much we actually understood about each individual

using the internet now it's I'm sure orders of magnitude more knowledge about each of us online so anyways that was the Inception the the uh the plan was to to take zero knowledge proofs we used the Z cash and at the time Komodo um you know Fork zero knowledge proof technology we've upgraded that since um we've got apologies for the interruption that was that was uh before sapling or or started before sapling and then we were actually the first uh chain to upgrade to sapling and we pushed all of the sapling upgrades back into the Komodo ecosystem which enabled uh pirate and and these other coins to to Really leverage that technology you know with the sapling advances and so that's been at the beginning and we've kept that as part of our plan and we we did a vision paper a couple months after the launch of mainnet actually um which described this concept of confidential verifiable but transparent polls in elections and the idea is that we described machine learning integrating with these provable confidential verifiable and transparent polls the idea is that if you if you provide a way for people to interact with the system and you give them absolutely zero motivation to lie through confidentiality this is a model for creating honest data from the world from the world of people and we've since had uh people join the community companies joined the community but one particular identity service that is doing basically identity as a service and it's a it's a way of bridging this completely Anonymous or or private um kind of blockchain with zero knowledge uh technology to the world of centralized systems kyc in this this company is leveraging the varus ID technology which we'll talk about later and I'll stop pretty quickly to um to be able to prove your ID in a way that would allow you to go to a company that requires kyc and say I'm kyc through this company they don't leverage any of my data they don't sell any of my data they don't disclose any of my data unless the specific reason for law enforcement requires it from them aside from that I'm kyc'd with them you don't need my data if you have any law enforcement requests you can get it from that from them other than that I don't need my data part of the sales and the you know the personal data Marketplace and so basically our whole entire ethos and what we've done as we built is you know a fair launch there was no pre-mine there was no Ico there were no initial investors Founders actually mined and stake their coins just like everybody else with a with a pre-announcement to the world and then us Founders we donated coins to a foundation that was intended so we donated our fairly mined and staked coins and a lot of people have since donated um to that Foundation but the foundation is private it's not a part of the public thing and and all it has done for the last five years is use all of its funds it's actually started with about 4 million it's got quite a lot less now and it's been basically making those funds available to the community that's when we were in Austin at consensus people donated to the foundation and then the foundation basically sponsored these things um through this kind of fundraising and donation process and that's the way we've done things for over five years now and uh and so Biz is actually he is in the ecosystem and he's one of the you know key developers in the

ecosystem but I'm an individual of a community I'm the lead developer of the community because that's it's like an honor to be the lead developer of the community if someone comes along and they're going to do you know more and happy to talk to them about having that role it doesn't matter to me as much as we're building something that's super important for the world it was designed with AI in mind it's really interesting that we are I believe weeks away from the final release of everything we've described in our vision paper and Now's the Time more than any that we need this technology as we see AJ emerging that's me that's the background that's how I got started in Varys I'll stop and turn it over to Biz because he's going to have a lot of things I think to say about what they're doing in him and what chips is thanks sure all right yeah thanks Mike um so I I'm essentially lead blockchain developer for chips um my background is I founded a fork of Monero called Blair back in June 2018. um how I got networked with Komodo and Mike uh is that I was approached by James Lee the founder of Komodo um and he basically asked me you know can you bridge compatibility between monero's code base and rdpal system so I started to work on that in about April 2019 um that's still underway I'm doing a lot of different things so it's been kind of hard to you know find time to push that across the finish line but yeah you can just briefly for uh for disambiguation John John just so people know John Lee is more commonly known as JL 777 yes yeah James Lee but yes correct James thank you I probably I only know him as JL 777 which is why yeah I got I got thrown there for just a moment uh but yeah sorry go on um you're fine yeah yeah so I I spent a few years uh kind of bridging compatibility you know we've we've notarized to komodo's mainnet uh a couple times from from a blur test Network um from there I wanted to get more familiar with the system became a notary node uh operator at Komodo I still hold that position I'll be running again this year for next year um and you know through that Network I was approached by satinder gruel uh he's one of the Komodo guys um you know one of the primary Motors behind you know uh ships still being project uh you know and migrating to varys's ecosystem so um how I got networked with Mike was essentially you know hey we have this um uh interesting case of chips which you know is a blockchain that's already reached maximum total Supply uh therefore in its you know current form mining was largely out altruistic block rewards have reduced to zero and basically the only people keeping the network moving which is a legacy Bitcoin code base um are doing so out of Goodwill which is not a great model right so um we also haven't had price Discovery occur and uh between the roadblocks we were seeing on the technical side you know those two factors uh Varys seemed like a really unique solution for us um in taking a snapshot of the existing chain and migrating that over to Varys we not only gain uh an entirely new fee economy and additional chain security that comes from that merge mining um we also have you know price discovery that can take place uh through varices liquidity pools and cross-chain bridges Gateway converters that kind of thing so miners will actually you know get revenues they would not be seeing outside of just fees being paid in chips and

it gives us you know access to all these D5 tools privacy features all the stuff that you know if if you're playing decentralized poker games which is what chips is uh you know what our current mission is to do um absolutely completely critical things uh so yeah we have since you have that up here we'll just talk about it um essentially chip started off as uh cryptographic exercise for uh JL 777 and sg777 uh SG is the primary motor behind the game engine so I oversee the blockchain development um the chain migration over to Varys I've contributed some code back up to Varys to allow us 10 second block times um those are fully configurable and have allowed us to have pretty high velocity gaming um each turn takes you know around 10 seconds um and we were able to have been in our original dependence on both websockets and lightning network using varus's tooling so if you scroll down a little bit you'll see some of the uh the GUI front end there's just some things about about chips yeah so there's our gaming interface um that's still under heavy development but you know two-way games of Poker are at least working presently um I know within the start of the interruption within the nft community there are several uh there are several trade groups uh among nft collectors that uh that are uh that are gambling focused right so and that are poker focused so you it turns out I mean and I wouldn't have guessed this two years ago I wouldn't have foreseen that this was a trend right or a meta within within the crypto and and uh and investing uh scene but apparently there's nft collectors who hold regular poker campaigns and they do so based on trait groups of nfts they've collected and I mean what what stronger narrative could there be for them than to say like yeah no we we we uh we contribute back to the ecosystem by using a fully decentralized and permissionless and probably Fair uh ecosystem like like what you're building here right absolutely I did I didn't know about that that's actually really cool in in one thing interesting is that um with the ethereum bridge to Veris which is it's been running now for quite some time on the test net and it'll be rolled out with the with the pbass release that's that I mentioned is just right around the corner and you can send nfts from varus to ethereum and from ethereum to Varys and to any pbas chain which is public blockchains as a service and chips is a pbass chain and so basically uh people could you know Biz I don't know if you have thought about this at all but people could send uh you know their crypto pumps whatever nfts they have they can send them over to the varus ecosystem including to the chips chain so you know maybe there'd be a later version that would allow people to play with their nfts as well definitely yeah some something we're looking to expand to eventually you know this is a longer term thing but uh using various IDs to allow people to actually monetize you know their own player identity um and they can launch tokens they can do all these different things with use leveraging various IDs much the same way that uh you know we have tokens on the chips blockchain called cashiers players things like that that are used for namespaces for bucketing of information publishing information so that subscribe drivers can receive that information the whole game Flow is based around this various ID

feature where people can publish their information to what's called a content multi-map on their ID and that's available on the blockchain the moment it hits the mempool so it allows for a lot of you know um highly interactive gaming on varus it's pretty interesting nice yeah that's amazing so and I apologize some Interruption again like it's just it's crazy you mentioned there's this Confluence of just cool ideas that have been that have been fighting to to get free over the last couple of years and I just think it's interesting that right now you happen to be nearing this uh you know this release candidate that that would completely fit the needs of apparently what I didn't realize were thousands of nft holders so you got this this highly interested group that would love to stay in uh in a decentralized economy um and here you were building that I mean I know you're building in plain sight I just wasn't paying attention yet well you know that's the thing is that we're building and we're not out there like we have a large community and people think uh when people go out and they talk about various oh you know they're they're showing something but it's just a large community city of people and the number of people that you actually see on Twitter you know posting things it's people from the community it's not they're not there's no one paid to Shell or anything like that it's just people from the community who really believe in what we're building so yeah we've been building and not really out there because I don't have time to go out and describe everything just like you know in the early days of Bitcoin Satoshi didn't have time to explain it to uh to certain people who have started systems since but um you know it's it's kind of the same we don't we we're building I mean we're we're you know I wake up I've been waking up for five years and building I think Biz I've seen him you know day and night at times just building away that's kind of oh yeah oh yeah there's an open mic here that's why when it comes to when it comes to building towards privacy or when it comes to building a bare minimum if you're contributing significantly to the to uh to the Foss side of the ecosystem because let's be real if if um and I apologize I'm going to take it take it your word I don't mean to doubt but take it your word because I haven't verified yet that that you uh basically you were able to make the post the sapling update to Z cash something that the rest of the Komodo ecosystem could use right wait just me saying that when there's a contribution to the to the free and open source movement uh and and new functionality is pushed into the ecosystem that's precisely why I do these sessions is because I want to reward good behavior that's what I'm saying um we need more of it um we definitely need people building significant uh privacy updates to existing code bases um allowing for for better peer-to-peer peer technology with uh lower metadata leakage and lower data leakage bare minimum to preserve some anonymity and then also cryptography right yeah we do need to tighten up our systems yeah to just take that a little bit further you know you're right it's so we pushed that technology back but I think it's much much bigger that you know chips can come along and say here's some infrastructure that we could use we you know uh Biz added the variable block time and that was just went back into

the various ecosystem so when people come and they work on this ecosystem they do what you're describing and this ecosystem enables anyone any project anyone who and people will figure it out and the people who figured out earlier like chips are going to benefit to come and launch a chain or launch currencies with zero knowledge privacy support with all of the D5 Mev resistant D5 by the way like we literally have a protocol that solves Mev it's the only one I know about and we could talk about that but we won't have time so it's we can do it a different day or something or or online I'd love to have people understand no I I appreciate that too because it's one of those things we could do a Roundtable on it too there there's so many groups that that need to be discussing Mev because it's well past minor extractable value right the narratives have shifted because finally we acknowledge that the problem is way deeper than Mining and so when somebody starts a chain they don't pay us for doing that they launch it there are fees on the blockchain that goes to Miners and stakers on the blockchain the chain launches it merge minds and leverages all the security and hash power of the main Network plus it's got the 50 proof of stake that was there from the beginning and that chain is now its own Community or it's self-sustaining it's Bridges self-sustaining it can communicate with the rest of the crypto ecosystem and so that's fully open source software that's yeah that's not rent seeking that's actually what we need as humans to be able to build out this infrastructure for the world without having some random person thinking that oh this is my chance to take all the money because that's that's just not how it's going to work I believe sure well let me ask you just uh to to draw a comparison to the IBC ecosystem which is which has become fertile ground for some interesting projects to right to start growing um but and you know you talk to like the cosmos maximalists because it's a thing now um and they say nope IBC is the only way because of uh because of the the atomic swaps that are built in as a primitive um they've had an issue with tokenomics right and the gas fee Marketplace for for the atom token itself Cosmos right they um it doesn't make okay I I I prefer not to um probably focus on a negative of another project but what I'll do is I'll say I can say uh my perspective of this so yeah when I meant it I didn't mean to no no only to ask yeah in terms of uh in terms of the ways that you know maybe various has solved that problem for your ecosystem yeah why would a developer when they look at the atomic swaps as A Primitive in that ecosystem and maybe that's an attractive thing for them why would they consider that Varys might be a better choice for them well see here's the thing Varys we just don't talk about it because Atomic swaps are like not much we have Atomic swaps built into the protocol and we have for a long time like for a few years we've had a Marketplace on chain that includes the ability to just did a decentralized way buy and sell IDs and anything connected to them and on the test net which has been out in an in release candidate mode now for a while um you can do that with all currencies so you and it actually leverages one of the original things that Satoshi put into Bitcoin you know it's it's uh it's just built people have

had this ability to start building stuff like this for so long and instead they followed a Smart Genius young guy who said let's just not do any of it let's just make a touring machine over a blockchain and let the world solve all our problems and it's like let's just build all the infrastructure that runs the internet on dos it's kind of the same idea on you know an operating system with no security um and that's what the world is kind of plowing forward on because everybody thinks that you have to move fast break things and take all the money yeah not not with people's not with people's Savings please don't move past the great things yeah and so and so what we we've been building we've been growing and we've been growing steadily and we don't I don't like I'm not here to predict any Financial things or any of that stuff but I mean we've been getting stronger and stronger we just announced that the Irish government is working with Dell and VMware um with with the varus protocol and uh one of the companies in the various uh Community archetype also known as value that's the company with the ID technology I was telling you about and they're doing uh a roll out of a sustainable energy currency that isn't just like carbon credits where you know that it was produced or there's actually a lot of green washing and fraud going on there but it's not just like carbon credits in the sense that it comes from uh some sustainable production we're tracking the production from all the way from you know whether it's Wind Farm or nuclear or whatever it happens to be that they I'm not on the policy side at all I've just been helping with the technology they can produce currencies from these sustainable produced uh sustainably produced sources they can prove by metering and regulation they can prove that that was consumed by say for example a Dell Data Center and the currency that's represented by that is exactly produced and consumed they also have the ability to track just produced and and where it was burned and you know the coolest thing about it and this is a guy in our community um consilience he's also uh the person who is behind that archetype company and and he brought this concept which they're doing which is if you have they don't have great storage for sustainable energy so you've got gigawatt hours of energy per day that is going to be wasted yeah my content stream all started with proof of work so I'm I'm very familiar with opportunities so because varus is still fpga resistant and you can the best miners are cell phones that are you know low cost low power cell phones because they actually the algorithm it's not that it's just never been implemented by an fpga there are fpga implementations it's that I designed it specifically in Assembly Language with instructions focused on the Asics that exist in modern chips today so that you can't beat it with an fpga and so you could run you know it's just not the best device to mine and so when you think about all this extra power and you think about data centers that are leveraging this technology to gain you know uh green currencies and things like that for their investors or for whatever else they can also use that to mine all of the thousands of blockchain networks on the pbas network across the world and turn that into a baseline price for that energy even if it's not used for other purposes so it's really just the whole

model kind of comes together in a circular economy for energy and they're doing it with the Irish government Dell VMware and archetype and so yeah it's not just a you know about making a token with tokenomics so if you look at Atomic swaps we have them on the Chain it is like one of the least powerful things that we have on the network so if we talked about them like other teams do saying they're the greatest thing we would just not be able to talk about all the things that are way way more important and so um I think the only reason it's become a trend in the last couple of years has been because of D5 summer the first D5 summer right and and uh ethereum benefiting most from the first D5 summer throw things out there to lose money and it's going to lose Grand Grandpa's money or Grandma's money or you know the people who just got into crypto and they don't worry about that before release it we've actually been testing what we have on testnet now for about three years we've been running defy with Mev resistance for more than a couple years but we're not focused on just let's take our let's take all the money we're focused on making this something that will last just like Bitcoin lasted but solve the problems that people need to solve that's why as far as I know we really are the only solution for Mev in the protocol watching people in the varus ecosystem merge mine six chains at the same time or as many as they want to at the same time be the engine behind processing defy transactions on a utxo code base where 50 of the coins are generated by proof of stake and the other 50 proof of work has been really cool I don't know that that exists anywhere else in crypto yeah no I don't think it is very unique that's a very special thing I mean I I try not to use that word too often but it is a very special thing um and it is it is sort of the hope of most people who maybe who aren't even interested in what you're doing yet like uh like I've been calling for more projects it's it's an ecumenical call to just consider merge mining more uh more of the tentpole projects right more the the blue chips so to speak this is all this is all Dynamic and built in so the new you know the new projects that start will be able to help Drive their launch by merge mining other projects and merge mining Varys and so by starting on this if you if you start a project you can get investment and you can start a project that way or you could start a fair launch project however you start it's actually an ecosystem that helps support you from the beginning it doesn't just extract rent from you from the beginning so if you're able to launch a project in this ecosystem the ecosystem itself helps your project to grow it's it's the way it should be and as far as I know there is no other there is no other blockchain technology and I believe I can confidently say this that's that is permissionless that scales to the world that's actually decentralized and that has a hope to provide a provable foundation for machine learning data from humans in any foreseeable future I only know of one okay got it well this is this is not their interview half hour so we won't bring them up I guess um but I think I'm intrigued of course Mike this is the first time actually talking I should ask before we before we go any deeper into the conversation for the benefit of the audience just let me briefly ask did I ask for a bounty in

order to set up this interview no you did not did you offer one we did not did did we discuss any kind of like compensation to have this stuff compensation that's it yeah so but actually actually I was thinking and this is uh do you have a night you have mine your Biz ID uh on virus I'm pretty sure I don't no we gotta get we gotta make sure you should have that yeah we're gonna squatter after this interview now that I've uh actually no this isn't live now I am self-interested okay I'm gonna I'll just admit I'm gonna be squatting on this interview until I go and secure that that uh I think I think you didn't ask for anything ahead of time we didn't offer anything ahead of time but I think it would really be fair as just a welcome to the community for us for someone to get you in mind your Biz ID so the first thing I'm going to do after this interview is go see about getting that for you well yeah I appreciate that you know what's funny is that I've got that I've got a various coin enamel enamel coin right physical enamel coin here but but I don't have any any other like actual meaningful exposure to the ecosystem that seems yeah a little bit silly now that I say it out the mobile wallet which is a multi-coin and multi-chain Mobile Wallet that was also fully open so it is fully open source and was developed by the community and Etc et cetera that's going to be on the stores uh hopefully within a week or so and it's available already it's been available actually for some years um for Android and iOS on test flight but it's going to be available on the main stores in about a week or so as far as package management is that available uh is there a a repository available for F Droid for those of us who are degoogled we don't we don't have a repository for F Droid we've talked about it you could if you if anyone makes I mean this is kind of the model the community development model is you know we're working on this pbas release so it's pretty much all hands that are available in the community on deck on release you know getting this pbass released on and we're hoping like I said that we're looking at a week or you know in that range kind of time frame for the main net release for all that um F Droid is definitely something that's that's interesting and we would want to do it but some of these things are probably going to come after the PBS release or someone could come along and you know give a PR okay yeah thank you yeah yeah thank you for this this ambiguation there I uh I work with the other kinds of PRS on a regular basis uh great that that's amazing um so looking forward to that there was one thing I wanted to add to your last question if I could it will only take a second sure you asked about tokenomics so we're not a token you know various is not a token there are plenty of models for tokenomics on the chain for tokens that people can create because you can create an unlimited number of tokens on the chain and if you do create one and send it over to ethereum it automatically just pops out as an erc20 with its Supply 100 controlled in a decentralized way by the varus protocol so you don't have to make if you wanted to launch ethereum currencies and tokens you could probably do it on varus easier than on ethereum and less expensively um but how many okay oh my gosh it's another Rabbit Hole how many standards are we talking about here well with nfts so any ID you can send it over

to ethereum and it pops out an ERC 721 nft automatically I mean these are part of the bridges so every single erc20 can be sent over to Varys and every single various currency or any currency on any chain connected to various can be sent over to ethereum um it just works that way and that's permissionless but the other thing about it and this is It's you could call it coinomics but it's just good protocol design all the defy uh and and IDs and everything else on any of the pbas chains every D file liquidity basket needs to have the native currency in it so on chips they're going to have D5 baskets they can have up to 10 different currencies but one of those currencies is going to be chips okay when you convert from one currency to another your maximum fee of conversion when you're going from one to another is point zero five percent the when you convert from one fee or from one currency to another the fees of conversion they go half to the liquidity providers half purchase the native currency of the blockchain and pay it as fees so it just converts automatically and pays it as fees so if your blockchain is used the miners and stakers will earn regardless of if you have block Rewards so this is kind of the point there's a there's something called a fee pool and it's like everyone talks about oh you know Bitcoin is in trouble because it's running out of rewards all they have to do is do what we did and they can follow us I'm happy to help them well ordinals kind of changed the game there too in terms of use cases there's from others who are figuring out that like oh yeah op return and Ops check checksig multi-sig uh that there's more opportunity there rather than this hidden opportunity for 10 years you still have a 3D minor problem though so you're right you you first you have to figure out a model for a fee economy that works that makes sense that isn't just trying to take money from people and then you need something like the varus feed pool where the way it works is we have a different model than other blockchains on how you take the fees as a miner or Staker and what you can do with them you capture all the fees in a block whether it's ID registrations or D5 conversions or transaction sends or any of these different things nft minting whatever it happens to be you take all those fees all of them and you put them into the feed pool which is a it's a number that it's actually just a an aggregate fee pool that continues in perpetuity on the chain and then you take 1 100th out for the current block so there's no motivation for any Miner or Staker to rewind and take the fees that you got because you got so many fees in that block and the next one doesn't have them everybody just moves forward all the motivation is about converging the chain and yet you can make ", "nextTranscriptToken": "RbgkyuNyQco 8 money based on activity on the chain so it's just solving the problem there's I don't think that there is a problem with tokenomics coinomics or any of that because it's all self-sustaining there's no company in the middle taking rent there's no one taking extracting anything and it's a circular economy designed that way I think it I I don't know of a problem with the design on this yeah that's pretty great what's yeah yeah and there are definitely examples of other ecosystems where they're where there are new problems cropping up every every

cycle right every uh every bull cycle we just discovered that there's some other other problem because the incentive is either unpredictable or some new um some new exploit becomes available or obvious uh for yeah for for some kind of greedy Behavior whether it's greedy mining or it's uh you know or it's some other yeah some other Behavior that's abusive right from the the staking or the delegation or yeah black ordering yeah and actually greedy mining I think generally comes from uh chunky fees and you know rather than like EIP 1559 the idea that oh we're gonna burn currencies because then we're going to make a super stable currency um yeah you know the idea that that you just destroy an economy that's growing in order to I'm sorry I decisions being made in crypto that don't have much forward looking in them and and it's it's good to see that you know we're getting recognized a little bit more people are starting to notice but by the time this is all available I think it's still going to take most people by surprise sure now I'm kind of curious in the uh in the various ecosystem or the various Community was there somebody that had uh that you know that has expertise in in uh in economies right that who is like a professional Economist and not that that has to be the measure but I mean what was kind of the The Guiding logic to to come up with some of those decisions that you mentioned just a moment ago well so some of us are um some of us have uh pretty deep backgrounds in a number of areas you know uh mine's more in the sciences and computer science and nanomaterials and all these things but it's really computer science and I just took a detour to invent some uh nanomaterials but um and now I'm back in software but we have uh we have um uh conciliences this is uh Nick Lyon Nicholas Lyons who's uh he was a I think a VP at JP Morgan but he was also a uh one of the principles at uh at a hedge fund and he brought in something like 92 billion in in uh Investments but he's really been focused on economic models and really kind of a broader picture for years like we we met in the early days of Varys and he had gone around he had talked to Roger very talked to Joe Lubin he had talked to a bunch of different people in the industry to try and figure out is anybody taking crypto where it really needs to go from an economic perspective and he um a mutual uh a mutual friend connected us and uh and he flew out from New York which he tells me never does just you know almost instantly we met and he decided this was the platform that actually was getting it right and he's been in the community and contributing ever since um so we have you know some like him we have uh other people in the community who do have a strong background in economics we have a one person I don't want to I don't want a docs because a lot of people in the community are just there but we have a one person who is um you know an ex lawyer from a large Financial uh firm who's really focused on on are these things fair I mean he's actually it's really interesting he'll say he's not a developer and he'll whip out blockchains with our protocol and he'll start up currencies and he'll do all this other stuff but he's not a developer he's got a legal background and he's got you know this this other type of background and and he'll come up he'll in the past he doesn't do it now because as far as we know

we got it covered but you know he would ping me in the early days when it's like okay we're Mev resistant because we solve everything in parallel and there is no front there is no back we do that and this is the only blockchain that does that the only because like I wrote the code to do that I don't know of any other code that does that um and and he says okay but if I'm mining and I want to front run somebody puts in a million dollar order for something could I just put my front run in my block and exclude everyone else so that it has to go into the next block and it's like well yeah I mean that's you know inter block uh you know Mev you could conceive of that and then it started me thinking and it's like okay I could do something about that you know and now we have multi-block uh Mev uh resistant protocols so you know you can't even know that doing that is going to allow you to frontrun anybody so yeah so it's you know we have people in the community and and we have people from all different walks of life because we're not a company we're a community and people come and they'll and they'll poke at something and we have a kind of a an ethos where if somebody comes in their just you know a troll and rude we don't we don't really take a lot of time with them if so yeah well there's there's not much time to spare for haters yeah but if somebody comes and they've got legitimate issues and they poke on something like we want people to do that find something wrong and come talk to us about it we're open we want to understand you know if we can improve because as we know as we see it right now this is the protocol that we want not as people developing as I want this for my family my you know I want this for everyone using crypto we got privacy we've got full decentralization and self-sovereignty nobody can take your funds nobody can take your data if you don't want them to you know it's yeah and let's do this pardon the interruption let's let's do this on the remaining the remaining remaining a few minutes of today's uh of today's session let's talk a little about varus ID because this is one of the more visible aspects like that that is sort of consumer facing let me uh bring up for three years yeah so let's let's talk about that uh various ID who who's using it why why would somebody come to this as a you know on the consumer side great question so all right I'll give you a few things first we have tens of tens of thousands I don't know the exact number registered on the Chain but they're all permissionless and all the ones that are on the Chain now are permissionless it is possible with pbest to create a permissioned ID namespace um as an option but on the current main net um the root IDs on any blockchain are the IDS with the most the broadest capabilities they can launch currencies they can launch other blockchains um and so these are the root IDs that are available now on mainnet they've been available for a few years uh they when you get an ID you have the ability to have data in your ID you have the ability to have a zero knowledge endpoint for your ID and you have the ability to register your ID with zero knowledge transactions that leak nothing about who registered the ID you also have the ability to use a referral which allows you to both reduce your price of registration on the blockchain which goes to minors and stakers and also when you use a referral there are

three levels of referral it basically if I refer you then if you refer someone else and that person refers someone else the person that you referred gets 20 from the referral they had you get 20 from the referral I get 20 from the referral and the rest goes to the miners and stakers on the blockchain and there's no again no Center no company taking anything and so these are options when you launch a blockchain you can say I don't want referrals on my IDs on my blockchains or you can say I want five levels of referrals on my blockchain you know and and you can do the same when you're launching liquidity baskets because liquidity baskets can also uh register IDs in in pbas so varis ID gives you this permissionless fully self-sovereign way to create an identity that has a friendly name you know any name you choose and it's a namespace so you can have the name you choose as a root name or you can have sub names Dot name okay now you also have the ability in there to have a z address for private transactions holding private funds receiving private messages and you have a public friendly name so if I want to send to a private address of say business um uh ID you know I can say send currency whatever I want to send or I could use the GUI but I'm you know I can use a command I can say send currency you know from whatever ID I have or maybe a wild card to Biz at colon private and whatever his uh private ZK snark address is that's bound into his ID it goes there and no one sees any metadata it's just it just it's just private transactions back and forth between IDs but there's a lot more you can do because IDs also have the ability to lock themselves so none of the funds they control can be spent until they are unlocked and you wait for a certain period of time multisig capable as well they're multi-sig up to 13 currency is cross chain capable and there and yeah and you can have IDs that Co they they appear on ethereum as ERC 721s and they appear on other pbas blockchains as their ID so you could have an ID that actually is across many different blockchains the exact same ID with the exact same name and everything else used on an unlimited number of blockchains and here's a really interesting thing about it so you've got this you got the ability to lock it but you also have the two other authorities on it that are also IDs you've got revocation and you've got recovery and they're also all multi-sync and and revocation can't spend your funds but it can revoke your ID so you could give that one to a fraud detection company if you wanted to and you're still safe recovery all it can do is recover the ID it's really pretty powerful but it needs to be revoked before it can do anything and it can't spend your funds either okay so you could share your recovery among your friends you can do all the every single thing that ethereum is just realizing they might be able to do with the account model we've been able to do but we're we still don't think like when PBS is on mainnet we believe that's the time we have time again now to go out and tell everybody about it and that's when everyone's going to I think be quite surprised at what they need is already built um but the other thing about IDs you can actually create the equivalent of an nft token from an ID but it's more than that it's like a super token which is an nft that's the thing that can pop out on

ethereum as an erc721 one token to rule all your IDs across all your chains really so you can send that token to any chain because Varys is inherently cross chain you're never bound to a specific chain with anything that any asset on the chain so you can send that token to any chain and you can send your ID to any chain so now you put your ID on all these different chains and you have different assets on those chains that are on that ID okay you can actually uh you could revoke your IDs on those chains transfer that one token and in one transaction you can transfer ownership of all assets on all chains where that ID exists with one single transaction guaranteed foolproof one transaction that basically made a transactional operation across every single chain where your ID exists so those are a few reasons and oh and one one last thing every ID has its own built-in unlimited size database that has create read update and delete operations as part of it even though blockchains are immutable and ours are too so your ID itself and this is uh what Biz was talking about actually has the ability for you to store an unlimited amount of information bound into your ID provable so like Elon Musk was on Twitter one day and he said oh it's so great that they're you know they're meeting like 22 000 tweets per second I did the math on that if I wanted to do 22 000 tweets per second on the blockchain yeah it's you know a couple of hundred 200 blockchains not a big deal like we literally don't have any limit on scale if you just use multiple chains for scale and apps don't have to worry about cross chain if all they care about data because you can just look at every chain instantly on the data that someone put in anyways those are IDs uh not really in a nutshell but those are IDs got it yeah I mean it sounds like like I know I'm gonna need to do a second Pass of All the documentation just to fully uh be able to internalize everything that you just mentioned about the various ID system because it sounds like I mean it's it's so Broad and so deep um it isn't just a road napped either but with so much it's in production right now yeah it's done and and one other thing the the mobile wallet includes uh various ID login without passwords so it's a fully cryptographically protected secure 2fa login that already exists we already have the protocol it's already in the wallet and we had someone who actually did a bridge to oauth 2 and uh open ID connect um which eventually will make available as just open software as well but using them really has the potential just end passwords and make everything more secure in the process that's awesome well let me ask as far as Hardware signing devices is that is that part of the security Paradigm for various ID and and for and for all of the uh part pardon uh the expression but just managing accounts and managing uh yeah managing all kinds of um you know wallets and so if you don't if you don't have IDs then I think Hardware uh wallets are really important if you do have it so let me just give you an idea so um if we definitely have designs and in fact someone was making a hardware wallet but we have things that we want in Hardware wallets that go just beyond simple signing of a transaction um you can certain that you can certainly store your keys in a hardware wallet and what I typically do if I wanna if I have IDs that I really want to secure

is I just I put the funds on the ID I locked the ID I put the revocation on a steel wallet and I stick it in cold storage and so I have uh live funds that can stake and earn money for me but they're as safe as cold stored funds you know so I I don't actually think like if you if you're generally using IDs Vault locks and things like that I don't know that you get a whole lot more security with any of the existing Hardware um Solutions but people like them and it's a kind of easy thing to understand and uh yeah it'd be nice right yeah I'm not I'm not a security expert I just based on what I've read and kind of best practices in in that domain from the people who I consider experts they talk about having or something you know something you have and something you are ask every expert who tells you that they that about this if they understand that various ID exists because it's a technology that never existed before and everyone who says that you need a hardware wallet as far as I know actually just doesn't understand it because I my my funds that are on my IDs where I've got a steel wallet hidden away somewhere and safe for example um you can't access those funds any more easily in fact in fact there's no chance you could fish me with you know a fishing Ledger you can't you can't give me an upgrade you can't do anything that's going to get me to sure but you're just you're describing you're describing cold storage of any network the use case you just laid out though Mike is is the same as cold storage for any network for example if I use it if you think you're right no you're right yeah but what you just laid out is is equivalent to me using as to me using uh wallet derivation software on an air-gapped computer and then imprinting a seed phrase onto a steel plate similar security similar security but let me tell you what you can do so I can still stake with all my funds you can't do that with cold storage I can actually prove my funds I can unlock my funds for a few minutes I can reset my my uh private my private key every time I unlock I can unlock for one minute spend my funds relock with a new key every time that's safer than I mean that's as safe as cold storage but I can still use my funds and I didn't need a hardware wallet you know what I'm saying okay yeah so I know that I know I need to do a little more digging into documentation to start like really grab my head around it this is the first exposure so well yeah but um but I'm hearing a bit I am picking a bit of what you're throwing down uh but there's a lot let's be very clear that there's a lot so this is uh this is the first uh first sort of uh sip for me and y'all gave me the firehouse but I'm doing this building we've been focusing on building so yes there's a lot I know yeah it's phenomenal so I think that concludes the amount of time that we can that we're we're gonna do in this episode mind your business audience and uh be loud in the comments yeah Biz go ahead uh just as a developer coming to Varys ecosystem solves so many problems for us all at once the amount of technical knowledge required to use their technology effectively is high but join their Community come around start building you'll find a lot of helpful people and suddenly the wall won't see so big so join up and uh you know contribute code to them they'll help you out get involved awesome awesome and my university community if you want

to if you want to hear more from these two gentlemen on a future episode here on mind your Biz leave comments be supportive uh and then you know you you tell us that this is this has been helpful for you in the meantime uh Biz you you that's the the call to action is for developers to join in and help build more Mike any other calls to action or upcoming dates or events that that the community should be aware of where support is needed and in what ways yeah everything I've described is fully functional fully operational we are not aware of any issues with it and it's running on testnet for people to come and develop with and use what Biz said I think that's true on the developer side but I'd say one more thing uh we have people developing solidity and developing on varus I think anything that you might decide to develop as an application and Biz please correct me if I'm wrong feel free would likely be easier to build using various technology than if you were to write it or try or attempt to write it in solidity I believe that's true I think in most cases that is yes definitely so I think the call to action would be you know come to our Discord try all this out we do have non-programmers using this and being able to create currencies and being able to create blockchains and people will help you do that if you're a developer if you're looking to make a crypto project take a look at this first you owe it to yourself it's going to be good for you in the long run um and then watch for the pbass mainnet release in the next week or two I'd but they're as safe as cold stored funds you know so I I don't actually think like if you if you're generally using IDs Vault locks and things like that I don't know that you get a whole lot more security with any of the existing Hardware um Solutions but people like them and it's a kind of easy thing to understand and uh yeah it'd be nice right yeah I'm not I'm not a security expert I just based on what I've read and kind of best practices in in that domain from the people who I consider experts they talk about having or something you know something you have and something you are ask every expert who tells you that they that about this if they understand that various ID exists because it's a technology that never existed before and everyone who says that you need a hardware wallet as far as I know actually just doesn't understand it because I my my funds that are on my IDs where I've got a steel wallet hidden away somewhere and safe for example um you can't access those funds any more easily in fact in fact there's no chance you could fish me with you know a fishing Ledger you can't you can't give me an upgrade you can't do anything that's going to get me to sure but you're just you're describing you're describing cold storage of any network the use case you just laid out though Mike is is the same as cold storage for any network for example if I use it if you think you're right no you're right yeah but what you just laid out is is equivalent to me using as to me using uh wallet derivation software on an air-gapped computer and then imprinting a seed phrase onto a steel plate similar security similar security but let me tell you what you can do so I can still stake with all my funds you can't do that with cold storage I can actually prove my funds I can unlock my funds for a few minutes I can reset my my uh private my private

key every time I unlock I can unlock for one minute spend my funds relock with a new key every time that's safer than I mean that's as safe as cold storage but I can still use my funds and I didn't need a hardware wallet you know what I'm saying okay yeah so I know that I know I need to do a little more digging into documentation to start like really grab my head around it this is the first exposure so well yeah but um but I'm hearing a bit I am picking a bit of what you're throwing down uh but there's a lot let's be very clear that there's a lot so this is uh this is the first uh first sort of uh sip for me and y'all gave me the firehouse but I'm doing this building we've been focusing on building so yes there's a lot I know yeah it's phenomenal so I think that concludes the amount of time that we can that we're we're gonna do in this episode mind your business audience and uh be loud in the comments yeah Biz go ahead uh just as a developer coming to Varys ecosystem solves so many problems for us all at once the amount of technical knowledge required to use their technology effectively is high but join their Community come around start building you'll find a lot of helpful people and suddenly the wall won't see so big so join up and uh you know contribute code to them they'll help you out get involved awesome awesome and my university community if you want to if you want to hear more from these two gentlemen on a future episode here on mind your Biz leave comments be supportive uh and then you know you you tell us that this is this has been helpful for you in the meantime uh Biz you you that's the the call to action is for developers to join in and help build more Mike any other calls to action or upcoming dates or events that that the community should be aware of where support is needed and in what ways yeah everything I've described is fully functional fully operational we are not aware of any issues with it and it's running on testnet for people to come and develop with and use what Biz said I think that's true on the developer side but I'd say one more thing uh we have people developing solidity and developing on varus I think anything that you might decide to develop as an application and Biz please correct me if I'm wrong feel free would likely be easier to build using various technology than if you were to write it or try or attempt to write it in solidity I believe that's true I think in most cases that is yes definitely so I think the call to action would be you know come to our Discord try all this out we do have non-programmers using this and being able to create currencies and being able to create blockchains and people will help you do that if you're a developer if you're looking to make a crypto project take a look at this first you owe it to yourself it's going to be good for you in the long run um and then watch for the pbass mainnet release in the next week or two I'd

Welcome to Verus

The Future Is Now

Here is a collection of information meant to span several platforms and services in order to keep all of us more informed on what's happening across the growing, world-wide community. It will also help to connect us, allowing us to discuss what's being worked on, what can be done to help and as a way to look ahead in a more involved way.

Verus, latin for "True", combines all of the features we believe are important in a cryptocurrency, and provides a foundation for future development. Verus is a fork of Zcash and Komodo that leverages the Komodo platform, and we appreciate the contributions and support from those teams as well as the Bitcoin developers that created a foundation for us to launch new capabilities **in a system that supports Bitcoin and Zcash compatible transaction types as well as Komodo cross chain swaps** and dPoW security enhancements.

1. Verus Coin is a pure technology provider and does not endorse third party projects.
2. Be your own bank: **you** are responsible for securing your coins and taking backups.
3. **No VerusCoin community member, admin, or developer should ever, under any circumstances, ask for your private keys or for you to send coins to them. Please report any incident in the #community-support channel.**

The VerusCoin Community is YOU!

If you've been around long enough and feel a little skeptical or disillusioned from a few bad experiences elsewhere, it's totally understandable. We've all had them and we know quite well that when unscrupulous projects abuse words like decentralization, interoperability, or protocol level solutions, it can have a *"boy who cried wolf"* effect.

We believe that actions speak louder than words. Anything that may be considered hype can be backed up with functional examples. What we're doing here is genuine. We are trying to be the real deal, **just as Bitcoin started out, as an open-source, fair launched, no ICO, or pre-mined, or even dev funded project.** Despite this, a foundation has been established by and for fellow community members. There is always more to be done and the foundation regularly pays out bounties to community members that help to make the Verus vision a reality. In fact, just about everything has been designed to reinforce the community. Designed with efficiency in mind, only the miners and stakers are rewarded for securing the chains. This ensures bare minimum costs to the end user starting up a chain and the sleek UI removes the technical barrier.

NEW TO VERUS?

What we're about:

Put simply, Verus is much more than any single ordinary blockchain; more of an entire ecosystem of **interconnected blockchains that all operate decentralized and at the protocol level**. Verus introduces Verus PBaaS (**Public Blockchains as a Service**), a true publicly notarized blockchain as a service with an easy to use built in wallet UI, designed to make it so that now anyone can start

their very own full-fledged cryptocurrency so long as it can be funded or supported by miners. Newly created chains are just as secure as Verus itself and have the ability to be merge mined (up to **22x** at once!)

Benefiting both the user and the miners, self-strengthening the ecosystem and ensuring minimal fees (no room for middlemen to take a cut). Reserve backed currencies are also possible with **fractional reserve** capabilities and since interoperability is paramount, exchanging currencies in **cross-chain transactions** is as simple as sending coins from one wallet to a different coin's wallet. **Conversions** are handled by a built in market maker that automatically determines price based on a predetermined curve and issues orders fairly to all buy/sells within a block, with zero spread and the added benefit of eliminating the well- established problem of front-running, just some amount of slippage based on the net buy/sell.. With reserve currencies, buyers can make buys without even needing sellers (and vice versa).
Amazingly, everything is done at the **protocol level** on a decentralized network, meaning Verus and it's associated chains can't be censored or stopped.

Getting started:

Important links:

Website: *https://verus.io*
Wallet: *https://verus.io/wallet*
GitHub: *https://github.com/veruscoin*
Explorer: *https://explorer.verus.io*
Discord: *https://verus.io/discord* Check https://youtu.be/YVOfIMjRf30 if you only see one empty channel.
Bitcointalk: *https://bitcointalk.org/index.php?topic=4070404.0*

Max Supply: 83,540,184
Algorithm: VerusHash 2.1 PoW/PoS - 50/50
Block time: 1 minute
Proof-of-Stake (PoS) ROI is currently between 6% and 8%
Halving Frequency: roughly every two years
Next Halving: Current ETA: Jan 2025 (Block #3,381,840)
Halving Countdown: *https:/countdown.verus.io/halving*

VerusID

VerusIDs are true self-sovereign identities and aren't simply an ID system as much as a fully functional blockchain protocol. There is no business in the protocol, but plenty of opportunity for those who use what it can do for identity applications. Verus ID provides:

Quantum ready friendly crypto addresses on the worldwide Verus network -- VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum

attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID -> destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully decentralized -- anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers.

Revocable -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

Recoverable -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

Private - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

A Built-in Decentralized Referral Program, Enabling Natural Growth.

Verus IDs will cost Verus to acquire, 100 Verus per ID to be exact, which can be discounted to 80 Verus with referral of an existing Verus ID. The interesting twist is that all of the cost of an ID goes back into the network, either as referral fees, which are a way to get discounts or possibly even make money through referrals and built into the identity transactions themselves, or as mining and staking fees. No one besides people participating in the network as miners, stakers, or ID referrers take any proceeds from the cost of an identity. If you refer someone new and they purchase an ID with your ID as a referral, they will receive a 20% discount on the ID. In addition:

1. You will receive 20 Verus directly
2. The person who referred you, if there is one, will receive 20 Verus
3. The person who referred person b, if there is one, will receive 20 Verus, and
4. The miner or staker will receive the remainder of the discounted 80 Verus cost

As you might expect from looking at the fact that anywhere from 20 to 100 Verus goes to miners and stakers of Verus for each new identity once identities hit mainnet, some Verus blocks may have VERY high rewards for some time. The best thing about that is that regardless of how much the Verus blockchain rewards miners over and above the pre-determined coinbase reward, it will be as a result of the **on-chain economy**, paid for by people buying identities with **no inflation** of the money supply! If you or your friends missed the early days of the Verus launch, **you don't want to miss this** new opportunity to be mining, staking or referring now **Verus IDs are activated on the mainnet!**.

VerusPay

Blockchain-integrated payment gateway for accepting Verus Coin (VRSC) in a WooCommerce ecommerce store

This plugin extends *WooCommerce* on *Wordpress*, adding the ability to accept cryptocurrency payments in Verus Coin (VRSC) using either an on-store wallet daemon (best for VPS or dedicated hosting stores) or manually configured VRSC addresses (best for shared hosting stores).

When an order is submitted via the VerusPay gateway, the order will be placed "on-hold" while awaiting payment from the customer. The customer has a limited time wherein to send the payment and the store monitors the wallet/address to confirm payment received before releasing the order and redirecting the customer to the Thank You page.

VerusPay uses limited API functionality for Manual Mode, to communicate with the blockchain explorer in verifying payments and with the veruspay.io API to get up-to-date price data. These API's do not receive any private data either about the store owner, store, or customer. The only data sent to the block explorer API is the public/transparent blockchain transaction and address used. For VerusPay.io API price data, only the store-set currency is sent to retrieve the current fiat exchange rate for Verus Coin.

VerusPay installation instructions

Be a part of it all

Focus on Verus' social media presence has picked up with new, fresh looking Facebook and YouTube channels so check them out and feel free to post something or simply explore the content. There's so much going on, plenty to learn about.

So much so that there is an *open call to anyone and everyone out there who can submit graphical or textual content* that can be used to help support the upcoming mainnet launch.

A stronger effort from our already wide base of members to post and utilize social media tools to help spread the word and generate new interested users is something we can all do right now. There is no better time than now and getting fresh new minds in here to discover what is being done is the first step in getting the great snowball rolling. To those of us who truly care and believe in this project, then this message is meant for you.

Looking Ahead

Since we are on the verge of an historical mainnet release with capabilities that have the potential to revolutionize both the financial and technical sides of fintech and send shockwaves across many industries. There are still lots of things to be done but each day we collectively push closer and closer to making it a reality.

Social Media Links

Check out the brand **new** community website, community social media pages, new Veruscoin YouTube and Facebook pages for helpful how to or to just keep up to date with things. Speaking of how to, we can always use more helpful videos if any community members would like to take it upon themselves to chip in. The Medium page also has lots of helpful guides and there's always

something to Tweet about. Be sure to drop by the discord to meet the vibrant and helpful community in real- time, where it isn't uncommon to find the developers offering assistance. Where else can you find that?

We are all working together to realize something great. If you'd like to join us and be a part of that special something but don't know how to, it can be as simple as checking out the links below and helping other's to understand that there's never been anything in existence quite like Verus, a complete, easy to use, fully decentralized blockchain ecosystem, designed specifically to benefit all participants in one of the most low cost yet efficient in every way, end-to-end systems to date.

Facebook - *https://www.facebook.com/VerusCoin/*
YouTube - *https://www.youtube.com/channel/UC_-KCHBxaDwSgNMdE3LMThg*
Discord -*https://verus.io/discord*
Twitter -*https://twitter.com/veruscoin*
Medium - *https://medium.com/@veruscoin*
Reddit -*https://reddit.com/r/veruscoin*
Community twitter - *https://twitter.com/VerusCommunity*

Closing Thoughts

As a community project of just regular Joes, it can't be stressed enough how important each and every one of us all are. The true power of decentralization is putting power back in the hands of **YOU**.

Verus Information Page.

There's a lof of information about the Veruscoin project. We have gathered as much information and resources and bundled them into this FAQ, as to give a quick overview

`VRSC Wallet & data location` on different OS:

Linux GUI: `~/.komodo/VRSC`
Mac OS: `/Users//Library/Application Support/komodo/VRSC`
Windows: `%AppData%\Komodo\VRSC\`

CLI binaries location in Verus Enhanced Agama installation:

Note: All locations are relative to the installation location of Agama:

Windows Verus binaries: `resources\app\assets\bin\win64\verusd\` contains `verusd` and `verus`
Windows Komodo (and asset chains) binaries: `resources\app\assets\bin\win64\` contains `komodod` and `komodo-cli`
Linux Verus binaries: `resources\app\assets\bin\linux64\verusd` contains `verusd` and `verus`
Linux Komodo (and asset chains) binaries: `resources\app\assets\bin\linux64\` contains `komodod` and `komodo-cli`

CLI binaries location in Verus Desktop installation:

Note: All locations are relative to the installation location of Verus Desktop:

Windows Verus binaries: `\resources\app\assets\bin\win64\verusd\` contains `verusd` and `verus`
Windows Komodo (and asset chains) binaries: `\resources\app\assets\bin\win64\komodod` contains `komodod` and `komodo-cli`
Windows Zcash binaries: `\resources\app\assets\bin\win64\zcash` contains `zcashd` and `zcash-cli`
Linux Verus binaries: `\resources\app\assets\bin\linux64\verusd` contains `verusd` and `verus`
Linux Komodo (and asset chains) binaries: `\resources\app\assets\bin\linux64\` contains `komodod` and `komodo-cli`

Various links:

Website: Official VerusCoin website
GitHub: VerusCoin source code for the VerusCoin wallet, miner and explorer
Block explorer: Main VerusCoin Explorer
Block explorer: VerusCoin Backup Explorer
Block explorer: Explorer for all Komodo Ecosystem chains
Social Media: Bitcointalk
Social Media: Discord
Social Media: Twitter
Social Media: Medium
Social Media: Reddit
Overview: Chain specifications
Miner: Monkins ccminer for CPU (recommended) or GPU
Miner: nheqminer for Windows, Linux, and MacOS
Reported mining speeds: spreadsheet to compare hashrates
Mining pool: https://luckpool.net/verus
Mining pool: https://pool.verus.io/
Mining pool: https://zergpool.com/
Mining pool: https://vrsc.mcmpool.eu/
Mining pool: https://wattpool.net/
Mining pool: https://www.nlpool.nl/
Mining pool: https://vrsc.dev-codex.com/
Mining pool: http://vrsc.52hash.com/
Mining pool: https://vrsc.ciscotech.dk/
Mining pool: http://www.lepool.com.cn:8088/
Mining pool: http://verus.bcmonster.com/
Exchange: https://atomicdex.io/ (VRSC/any listed coin)
Exchange: https://app.stex.com/de/trade/pair/BTC/VRSC/1D (VRSC/BTC)
Exchange: https://safe.trade/trading/vrscbtc (VRSC/BTC, VRSC/SAFE)
Exchange: https://www.aacoin.com/#/trade?symbol=VRSC_BTC (VRSC/BTC)
Exchange: https://graviex.net/markets/vrscbtc (VRSC/BTC, VRSC/KMD, VRSC/USD)

Exchange: https://www.kuangex.com/#/exchange/vrsc_usdt (VRSC/USDT)
Guide: How to mine in under 5 minutes
Wallet: Verus Desktop releases
Wallet: CLI wallet releases
Mobile wallet: Android Mobile Wallet
Mobile wallet: iOS Mobile Wallet
Paper wallet: Verus Paper wallet
Bootstrap: Download Verus Bootstrap
VerusPay: VerusPay setup guide

Information:

Platform:	Komodo
Project type:	Decentralized public blockchain Open source, fair launch, no ICO Community driven project
Privacy:	Sapling enhanced, zk-SNARKs zero knowledge proofs
Website:	https://verus.io/
Github:	https://github.com/veruscoin There is the VerusCoin source code for the VerusCoin wallet, miner and explorer.
Block explorers:	https://explorer.verus.io/ https://explorer.vrsc.0x03.services/ https://dex.explorer.dexstats.info/
Announcement:	https://bitcointalk.org/index.php?topic=4070404.0
Discord:	https://discord.gg/VRKMP2S
Twitter:	https://twitter.com/veruscoin
Medium:	https://medium.com/@veruscoin
Reddit:	https://reddit.com/r/veruscoin
Mining Algorithm:	VerusHash 2.1 - PoW/PoS 50/50 = **Proof of Power - 51% attack resistant** CPU targeted, non-discriminatory, CPU/FPGA equalizing algorithm
Block time:	1 min
Block reward:	24 VRSC since 09/28/2018 (both mining and staking; next halving on Sept. 28, 2020)
Total supply:	83'540'184, no premine, no ICO.
Reward emission schedule:	Started with linear ramp, changing every block for the first 10,080 blocks (0 to 384 VRSC) and then halving every month for the following 5 months, then every 2 years after that. All rewards, at or above 192 are time locked for random release between block 129,600 and 1,181,520 (3 months to 2 years and 3 months after genesis).
Reward maturing:	100 blocks to maturity
PoS Period:	150 blocks cooldown of UTXO
Unique features:	VerusID: VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity

applications.

Mining:

VerusCoin can be mined with CPUs, GPUs and FPGAs, solo and in pools. However, the algo is carefully designed for CPUs, and they still substantially outrange GPUs.
ARM mining works in general, but although fairly efficient, without high hashrates. FPGAs can mine this coin, but don't outperform CPUs by much.

Miners:

Besides solo-mining with Verus Desktop or CLI wallet, you can use:
ccminer for CPU (efficient for most modern CPUs) or GPU
https://github.com/monkins1010/ccminer/releases;
nheqminer https://github.com/VerusCoin/nheqminer/releases (for Windows, Linux, and MacOS);

A spreadsheet to compare hashrates can be found here:
https://docs.google.com/spreadsheets/d/1RrSYJDV0Mjj3X-myMC3aQDGkcipivxHsD7ZxJ3r5f_A/edit#gid=201266774
You can also compare older and current algos

Mining pools:

https://pool.verus.io/ (fees will go to VERUS foundation)
https://luckpool.net/verus
https://zergpool.com/
https://vrsc.mcmpool.eu/
https://wattpool.net/
https://www.nlpool.nl/
https://vrsc.dev-codex.com/
http://vrsc.52hash.com/
https://vrsc.ciscotech.dk/
http://www.lepool.com.cn:8088/
https://zpool.ca/
http://verus.bcmonster.com/

Staking pools:

https://discord.gg/4FJJRY5 (Ginasis Staking pool, 5% fee)
https://docs.google.com/spreadsheets/d/1Up1WbMuCR21e6TxLae6zjLJePu_RIOVZwqShRj9vVvc/edit?usp=sharing (Dudezmobi Staking pool, 1% fee)
https://www.aacoin.com/#/trade?symbol=VRSC_BTC (Technically not a pool, but it stakes your VRSC on the exchange, 20% fee)

Price: https://veruspay.io/price/

Exchanges:

https://app.stex.com/de/trade/pair/BTC/VRSC/1D
https://safe.trade/trading/vrscbtc
https://www.aacoin.com/#/trade?symbol=VRSC_BTC
https://graviex.net/markets/vrscbtc

https://atomicdex.io/ (in Beta)
https://www.kuangex.com/#/exchange/vrsc_usdt

Also check out the Guide on how to mine in under 5 minutes: https://medium.com/veruscoin/how-to-start-cpu-mining-verus-coin-vrsc-from-your-laptop-in-under-5-minutes-f69c9aae340e

Wallets:

Verus Agama GUI wallet is a forked Agama Desktop App for multiple coins (for Win, Linux and Mac). It's been deprecated now and no longer supported;
Verus Desktop GUI wallet is a newly developed multi-coin wallet supporting VerusID
https://github.com/VerusCoin/Verus-Desktop/releases (for Win, Linux, Mac and ARM Linux);
the CLI wallets can be found here: https://github.com/VerusCoin/VerusCoin/releases/
There's also possibilities to test coming PBaaS functionality in this wallet.

A beta version of the Android mobile wallet can be found here:
https://github.com/VerusCoin/Verus-Mobile/releases
A beta version of the iOS mobile wallet can be accessed via this Apple TestFlight invite
https://testflight.apple.com/join/ZS43lYcw

Our Paper Wallet can be accessed here: https://paperwallet.verus.io

If you need a bootstrap, you can find it here: https://bootstrap.verus.io
(a guide how to apply is pinned in #community-support channel in Discord:
https://discord.gg/VRKMP2S or in the HOW-TO & FAQ section of our website
https://wiki.verus.io/how-to/how-to_bootstrap.md)

For those interested in VerusPay, a guide can be found here: https://veruspay.io/setup/
And finally for those interested in running various Veruscoin services:
https://github.com/VerusCoin/VerusServicesSetup

Information compiled by Thoskk.vrsc@, complemented by Godballz.vrsc@ and Oink.vrsc@

- Home
- Website
- Explorer

Release Notes

- v1.0.6
- V1.0.5
- V1.0.4
- V1.0.2

- Simultaneous blockchain and blockchain liquidity pool launches
- v1.0.1
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- What’s New
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Conversion Queries

- Defining a Currency

- Signing transactions from multi-signature IDs (testnet and mainnet)

- Tokenizing ID control (next generation NFT):

- Creating an identity with a fractional currency as its parent

- Verifying Downloads

- Mainnet Changes - Wallet Hardening

- What’s New for TestNet

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads

- Mainnet Changes
- What’s New for TestNet
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- Mainnet Changes - no protocol changes
- What’s New for TestNet
- The Path for PBaaS to Mainnet
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Conversion Queries

- Defining a Currency

- Signing transactions from multi-signature IDs (testnet and mainnet)

- Tokenizing ID control (next generation NFT):

- Creating an identity with a fractional currency as its parent

- Verifying Downloads

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads

- Verus PBaaS and DeFi RC6 - BUG BOUNTIES
- Testnet Reset
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- What’s New
- Verus PBaaS and DeFi RC6 - BUG BOUNTIES
- Testnet Reset
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- What’s New
- Ethereum Bridge
- Verus PBaaS and DeFi RC5 - BUG BOUNTIES
- Testnet Reset

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent

- Verifying Downloads
- What’s New
- Ethereum Bridge
- Verus PBaaS and DeFi RC5 - BUG BOUNTIES
- Testnet Reset
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**

- Conversion Queries
- Defining a Currency
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Tokenizing ID control (next generation NFT):
- Creating an identity with a fractional currency as its parent
- Verifying Downloads
- New Signature Capabilities on Mainnet & Testnet
- Verus PBaaS and DeFi RC5 - BUG BOUNTIES
- Emoji IDs and 🐼Chain
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- Platform Support
- PROTOCOL RELEASE CANDIDATE RC1

- NEW PBAAS CAPABILITIES
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency

- Verifying Downloads
- PROTOCOL FEATURE COMPLETE
- NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)
- NEW PBAAS CAPABILITIES
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- PROTOCOL FEATURE COMPLETE
- NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)
- NEW PBAAS CAPABILITIES
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads
- PROTOCOL FEATURE COMPLETE
- NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)
- NEW PBAAS CAPABILITIES
- Testnet Reset

- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Defining a Currency
- Verifying Downloads

- Announcing v0.9.2, OPTIONAL, HIGHLY RECOMMENDED FOR MAINNET, MANDATORY FOR TESTNET USE - THIS IS A BIG ANNOUNCE, BUT IF YOU CARE ABOUT THE PBAAS UPGRADE, WE RECOMMEND READING IT ALL.

- PROTOCOL FEATURE COMPLETE

- NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

- NEW PBAAS CAPABILITIES

- Testnet Reset

- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Currency Options
- Verifying Downloads
- Testnet Featuring Verus PBaaS
- Verus PBaaS Features Live on v0.9.1-2 Testnet
- Testnet Reset
- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Currency Options
- Verifying Downloads
- CRITICAL FOR NOTARIES, POOLS, EXCHANGES, MANDATORY FOR TESTNET USERS, HIGHLY RECOMMENDED FOR ALL MAINNET USERS
- Testnet Featuring Verus PBaaS
- Verus PBaaS Features Live on v0.9.1-1 Testnet
- Testnet Reset

- Signing transactions from multi-signature IDs (testnet and mainnet)
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Conversion Queries
- Currency Options
- Verifying Downloads

- Testnet Featuring Verus PBaaS -- The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

- Verus PBaaS Features Live on v0.9.1 Testnet

- Testnet Reset

- Signing transactions from multi-signature IDs (testnet and mainnet)

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Conversion Queries
- Currency Options
- Verifying Downloads
- This version fixes the following issues with v0.9.0-2
- New Command and API for Verus Vault - setidentitytimelock
- Verus ID and NFT Marketplace
- Verus Vault
- Testnet Reset
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER. IF YOU HAVE NOT UPDATED AND ATTEMPTED TO STAY CONNECT TO MAINNET, YOU MAY NEED TO BOOTSTRAP OR RESYNC YOUR NATIVE WALLET OR NODE AFTER UPGRADING.
- New Command and API for Verus Vault - setidentitytimelock
- Verus ID and NFT Marketplace
- Verus Vault

- Testnet Reset
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency

- Defining a Currency

- Defining a PBaaS blockchain

- Limitations in multi-currency to be aware of:

- Testnet Reset

- Verifying Downloads

- TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.

- Verus ID and NFT Marketplace

- Verus Vault

- Testnet Reset

- Exporting an ID to a PBaaS chain

- Signing transactions from multi-signature IDs (testnet and mainnet)

- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:

- Testnet Reset
- Verifying Downloads
- TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.
- Verus ID and NFT Marketplace
- Verus Vault
- Testnet Reset
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency

- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset

- Verifying Downloads
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency

- Defining a Currency

- Defining a PBaaS blockchain

- Limitations in multi-currency to be aware of:

- Testnet Reset

- Verifying Downloads

- Exporting an ID to a PBaaS chain

- Signing transactions from multi-signature IDs (testnet and mainnet)

- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain

- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Exporting an ID to a PBaaS chain
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads

- Exporting an ID to a PBaaS chain

- Signing transactions from multi-signature IDs (testnet and mainnet)

- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Announcing NON-MANDATORY, RECOMMENDED UPDATE FOCUSED ON NETWORK CONNECTIVITY AND ROBUSTNESS IMPROVEMENTS v 0.7.3-10.
- Signing transactions from multi-signature IDs (testnet and mainnet)

- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Announcing immediate availability of NON-MANDATORY, RECOMMENDED ESPECIALLY FOR NOTARIES, HIGH PEER COUNT PUBLIC NODES, AND POOLS, v0.7.3-9 Verus CLI.
- Signing transactions from multi-signature IDs (testnet and mainnet)
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Announcing MANDATORY UPDATE v0.7.3-8, which will re-enable full Verus Proof of Power, including staking, at block 1576200, occuring on Thursday at approximately 5:00PM UTC time. YOU MUST UPDATE BEFORE THEN TO FOLLOW THE MAIN CHAIN.
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain

- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs

- Conversion Queries

- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.
- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset

- Verifying Downloads

- New Verus Multicurrency, Multichain, DeFi Enabled Testnet

- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Staking-capable time locking and theft prevention (Verus Vault)

- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Consensus integrated DeFi liquidity pools and fractional currency baskets

- Simultaneous blockchain and blockchain liquidity pool launches

- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- An interoperable, multichain network for new use cases and unlimited scale**

- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency
- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- Mainnet Updates
- New Verus Multicurrency, Multichain, DeFi Enabled Testnet
- The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

- Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID
- Staking-capable time locking and theft prevention (Verus Vault)
- Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming
- Consensus integrated DeFi liquidity pools and fractional currency baskets
- Simultaneous blockchain and blockchain liquidity pool launches
- Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)
- An interoperable, multichain network for new use cases and unlimited scale**
- Locking and Unlocking IDs
- Conversion Queries
- Sending and Converting Currency
- Defining a Currency

- Defining a PBaaS blockchain
- Limitations in multi-currency to be aware of:
- Testnet Reset
- Verifying Downloads
- The Verus DeFi Testnet -- THERE IS NO TESTNET RESET
- Defining a Currency
- Verifying Downloads
- The Verus DeFi Testnet -- THERE IS NO TESTNET RESET
- Defining a Currency
- Verifying Downloads
- The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

- Defining a Currency
- Verifying Downloads
- The Verus DeFi Testnet -- THERE IS NO TESTNET RESET
- Defining a Currency
- Verifying Downloads
- v0.7.2-4 WITH THE WORLD'S FIRST FRIENDLY-NAME, zk-SNARK ADDRESSES FOR MAINNET AND TESTNET
- The Verus DeFi Testnet -- THERE IS NO TESTNET RESET
- Defining a Currency
- Testnet Reset Instructions
- Verifying Downloads
- Testnet Reset

- The New Verus DeFi Testnet
- Defining a Currency
- Testnet Reset Instructions
- Verifying Downloads
- Testnet Reset
- The New Verus DeFi Testnet
- Defining a Currency
- Testnet Reset Instructions
- Verifying Downloads
- Testnet Reset
- The New Verus DeFi Testnet

- Defining a Currency
- Testnet Reset Instructions
- Verifying Downloads
- Testnet Reset
- The New Verus DeFi Testnet
- Defining a Currency
- Testnet Reset Instructions
- Verifying Downloads
- New in v0.7.1-5
- Sending and Converting Currency
- Defining a Currency

- Testnet Reset
- Verifying Downloads
- New in v0.7.1-4
- Sending and Converting Currency
- Defining a Currency
- Testnet Reset
- Verifying Downloads
- New in v0.7.1-3
- Sending and Converting Currency
- Defining a Currency
- Testnet Reset

- Verifying Downloads
- New in v0.7.1-2
- Sending and Converting Currency
- Defining a Currency
- Testnet Reset
- Verifying Downloads
- New in v0.7.1-1
- Sending and Converting Currency
- Defining a Currency
- Testnet Reset
- Verifying Downloads

- Sending and Converting Currency
- Defining a Currency
- Testnet Reset
- Verifying Downloads
- The network upgrade activated the following features:
- New in v0.7.0-4
- Verifying Downloads
- The network upgrade activated the following features:
- Verifying Downloads
- The network upgrade will activate the following features:
- Notable Changes:

- Verifying Downloads
- The network upgrade will activate the following features:
- Verifying Downloads
- Notable Changes:
- Testnet Reset
- Defining a Currency
- Verifying Downloads
- Notable Changes:
- Defining a Currency
- Verifying Downloads
- Defining a Currency

- Verifying Downloads
- New in v0.6.4-3
- Verifying Downloads
- New in v0.6.4-1
- Verifying Downloads
- New in v0.6.4
- Verifying Downloads
- Verifying Downloads
- Verifying Downloads
- Verus ID
- VerusHash 2.1

- Making a new identity
- To create an identity:
- Testnet Reset
- Verus ID
- VerusHash 2.1
- Making a new identity
- To create an identity:
- Testnet Reset
- Verus ID
- VerusHash 2.1
- Making a new identity

- To create an identity:
- Testnet Reset
- Making a new identity
- To create an identity:
- Testnet Reset
- Making a new identity
- To create an identity:
- Testnet Reset
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:

- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- Here are a few examples of cross-chain operations using the command line:
- 0.4.0c - released 10/29/2018
- Version 0.4.0c also fixes the following issues reported in prior versions:
- For CLI:
- For GUI:
- Fixes:

- Fixed:

- Verus Coin 0.4.0 - Release Notes

v1.0.6

Announcing v1.0.6 - MANDATORY MAINNET UPGRADE WITH COMMUNITY AGREED DEADLINE ON TUESDAY, MAY 30th, 19:00 UTC

As everyone can see, there is a currency on the chain called RaceCondition, which someone has defined and is a 5% algorithmically controlled fractional reserve of Verus. While these currencies are tools, and there may be good reasons for 5% currency components or even 5% currencies in some applications, this was launched as a VERY low reserve (launched with 10 VRSC in reserves), high supply currency (100 million supply), and as volatile as the protocol allows at 5%.

The extreme nature of the currency and the fact that the person who launched it exited with their entire pre-launch amount when someone else joined resulted in a behavior different than what I would expect. Someone converted into the currency, which was clearly all in the hands of one individual. That individual then converted all of the currency they got at launch to VRSC, getting the highest price they could get for all of their currency and leaving the highly volatile, 5% fractionally backed currency with far fewer reserves relative to supply than it had before, basically an exit scam for a grand total of 4 VRSC.

The thing that made me take notice is that there seemed to be more reserves left and a higher price at the end of this operation than I'd expect in the currency, and that turned out to be because only in the case of a straight conversion from fractional to a reserve, not in any other case, the fractional currency fees that should have been burned into the basket were not being captured. Interestingly, in this case it resulted in a form of buffer at the extreme, where the person who entered RaceCondition for 5 VRSC after it was launched and subsequently had 4 of it taken on the exit from one person converting 100,000,000 of the supply ended up with more than they would have, had the math been properly distributing those fees. This was the totality of the issue we found, and though it may not seem that serious to require a pause of DeFi, these calculations are consensus, and we take any error

that we find in them, fees or otherwise, quite seriously. This update will operate on the updated rules when the oracle notifications are removed and DeFi is reactivated.

The fix means that all fees, including the fractional fees on an exit from a liquidity basket are burned before conversions take place, which in this case, would enable the person who exited with 100MM RaceCondition to take 15 VRSC, instead of the 14 that they did. This is because the correct fee allocation results in a 15 SAT price, which means the person who remains holding the RaceCondition bag would have only some 100s of thousands of SATs left in reserves. On the other hand, anyone could come in and reinflate the 5% currency by converting to it. This event on the network had the effect of reducing the extreme difference between supply and reserves on RaceCondition, and it will operate as a normal 5% reserve currency should when DeFi reactivates.

The lesson here is that if you’re interested to learn about a basket currency, pay attention to the total reserve ratio, what currencies are in it, and see that there is enough liquidity of reserves and holder distribution for the basket to make sense. Of course baskets could be memes, but they also have real function. Make sure you take a look into how a basket works and that you have a reason before putting your hard mined, staked, won, or purchased Verus in.

We will need the network to upgrade to version 1.0.6 before we can re-enable DeFi and cross-chain functions, making the ETH bridge launch possible, as well as PBaaS launches and conversions.

We see that someone is launching a currency called “whales”, which looks like an ID registration currency that will enable registering IDs with 3 levels of referrals in a currency that simply collects registration fees, which may start quite reasonably priced and likely get more expensive over time as fees are earned in the basket. The startblock for whales is 2561900, which should happen somewhere around 6:00 UTC on Thursday. That means that if we can upgrade and activate by Tuesday before the meeting, that person would still have time for pre-launch participation by those who would like.

Thanks everyone for upgrading as soon as you are able!

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The unlockatblock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"unlockatblock": <Unlock
block height>}'
```

- **Time Delay:**
  The setunlockdelay parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"setunlockdelay":
<Unlock block delay>}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128                   // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

  "name": "PBaaSChain",
```

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

```
    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",
```

```
      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults


```
  {
```

```
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
  ],
  "initialsupply": 3000000
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/74cdf4ce0f8a5735960f4b5ba0177080f94e9729f4f83a08133516d383518f9c/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/7eae55dc13e130d2b381e41309a5b3dc262b2326f2ae3a24fe358ace2431e697/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/26eb9299a6cb6f5d90e0f80a4eeba3e119563044c6133fa453e64a633c9bca3f/detection
Windows:
https://www.virustotal.com/gui/file/f93d13ba778d9b34641879d6c3d23a1bac001f3749c2756c651a3f9c9bb0d522/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Aa0DJwABQSChXw/NphqzkcaFyWwJzhP0g+78oxZoJtFIB8CR4J+X/DcQdXTR849aOvHYkCBc1UKoTmW
mrXXZs5iwjYutbcs3" "/Downloads/Verus-CLI-Linux-v1.0.6-x86_64/Verus-CLI-Linux-
v1.0.6-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Aa0DJwABQSChXw/NphqzkcaFyWwJzhP0g+78oxZoJtFIB8CR4J+X/DcQdXTR849aOvHYkCBc1UKoTmW
mrXXZs5iwjYutbcs3"
67cc123c25b7bf7ad01d16c1b9556f80c18fee3595f6e556d0245a67ef51263f
```

The result will be true if the signature is valid.

```
true
```

V1.0.5

Announcing v1.0.5 - MANDATORY MAINNET UPGRADE WITH DEADLINE TBD BY COMMUNITY (STRAWMAN SUGGESTION MONDAY, MAY 29th, 19:00 UTC)

While preparing the cross-chain VRSC currency launch and as part of continual review, we determined that one numeric function that is used in the cross-chain challenges and is calculated

differently for mainnet, due to the original launch that included the deprecated time locked rewards, was incorrect. This had never been hit, as it would have been calculated incorrectly only if multiple chains experienced a challenge of two competing forks, but in that case, it would falsely reject any challenge proof. That would mean cross-chain challenges on mainnet, even though they were fine and fully tested on testnet, could fail and block a bridge until that calculation was made correct and the network upgraded.

Because we have the notification oracle technology, we triggered a rip-cord notification selectively for cross-chain that made all nodes listening to the oracle, which we expect to be near 100%, disallow cross-chain operations without affecting same chain operations. That means that all functions on the Verus network, except launching PBaaS chains or Ethereum bridges are fully functional and unhampered in any way at this time. If someone tries to launch a PBaaS chain while this oracle notification is active, all nodes following the oracle will reject such an action without incident until the notification is removed.

This release, v1.0.5, properly calculates the aptly named “magic number” for each chain, in the correct way on the mainnet network, and as long as the notification is set, is fully compatible with current versions on the network.

Once everyone has had a chance to upgrade to v1.0.5, the network/community should agree to remove the notification from the oracle and allow PBaaS launches without false rejection of cross-chain challenges. Once we do that, and when someone launches a PBaaS chain that issues such a challenge, the network would become incompatible with Verus versions prior to 1.0.5.

From our perspective, we are not aware of urgent efforts to launch a PBaaS chain in the next few days by anyone who cannot wait. All things considered, we believe that we can at least wait until the community discussion on Saturday, and that a strawman deadline would be **Monday, May 29th, at 19:00 UTC** for everyone to have upgraded. If you are urgently wanting to launch a PBaaS chain, have a plan, and are prepared to do so sooner, we invite you to join the community meeting on Saturday at 19:00 UTC and let everyone know.

Having this notification remain in place until removed keeps the network running smoothly as it has been since activation, with all DeFi, multicurrency, and non-cross-chain functions enabled. Since the ETH bridge is undergoing final changes and review, is not quite ready to launch on testnet or

mainnet, and will likely require at least a couple days on testnet to finish its review, we do not believe the current notification is hampering or slowing anything or anyone down at this time.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised

a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may

continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The unlockatblock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"unlockatblock": <Unlock
block height>}'
```

- **Time Delay:**
  The setunlockdelay parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"setunlockdelay":
<Unlock block delay>}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals
```

```
OPTION_TOKEN = 32                       // if set, this is a token, not a
native currency




OPTION_SINGLECURRENCY = 64              // for PBaaS chains or gateways to
potentially restrict to single currency




OPTION_GATEWAY = 128                    // if set, this routes external
currencies




OPTION_PBAAS = 256                      // this is a PBaaS chain definition




OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices




OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain




OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

```
}
```

The bridge definition has overridable defaults

```
  {
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/c48816428e6baf6158eee0f553e587c7c7af73aa02ef7ffb4b84a2c3e72eba7a/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/7aa2923ce230e0cf96a981d5c1431b627aae2991958c380f64fffa6e7bbcabd1/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/691899be0d2cbdbd67cc58be042eec60e1f8b9f819347e026b3e556c9fc9a82a/detection
Windows:
https://www.virustotal.com/gui/file/6b2a03233a1eac9c27df66ee86a2490173d60248f55d5d2e35df1f1a442f6fcf/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AR/tJgABQR+VscwlZ89kaHfFayrfoMKgHCqMSLIXLTXYPCWuLbo4CTXU9zRogo75wjP2zr2nfqiWM6A
iUxVPm7oUjzmwovTa" "/Downloads/Verus-CLI-Linux-v1.0.5-x86_64/Verus-CLI-Linux-
v1.0.5-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AR/tJgABQR+VscwlZ89kaHfFayrfoMKgHCqMSLIXLTXYPCWuLbo4CTXU9zRogo75wjP2zr2nfqiWM6A
iUxVPm7oUjzmwovTa"
b6d9aebadf46e8310ba2f261f1865bb54bfd96c86ef7f9922ed073cde1811cb6
```

The result will be true if the signature is valid.

```
true
```

V1.0.4

Announcing v1.0.4 - CRITICAL, NON-MANDATORY MAINNET UPGRADE
THIS UPDATE IS CRITICAL FOR MAINNET DEFI OR CROSSCHAIN WALLET FUNCTIONS, AS IT RECTIFIES POTENTIAL REFUND FAILURES IN EDGE CASES - IT IS NOT REQUIRED FOR MAINNET SYNCHRONIZATION OR MAINNET FUNCTION PAST PBAAS ACTIVATION, BUT IT IS HIGHLY RECOMMENDED.

DO BE SURE TO UPGRADE TO v1.0.3 OR LATER BEFORE MAY 23, 2023, EXPECTED BLOCK #2549420

FOR CONTINUED TESTNET USE, UPGRADE ASAP

Mainnet improvements:

- Properly inserts refund addresses in all cases when making DeFi or cross-chain transactions
- Merge mining improvements from @Hellcatz for higher merge mining performance on all platforms and full pool merge mining support when used in combination with @Hellcatz and @Oink70's SNomp pool improvements
- CLI now defaults to mainnet for all PBaaS chains unless -testnet is specified
- 2 versions of GUI will be released for now, a mainnet and separate testnet version. Full PBaaS support in the wallet will only be for either mainnet or testnet, depending on the version used.
- Extends deprecation height from the previous 20 weeks to 52 weeks

Testnet fixes/improvements:

- Separate GUI release for testnet

- Refund address fix

This the is last expected and recommended release before the mainnet activation, Tuesday at block #2549420, which will add some pool merge mining improvements. Please make sure you update to v1.0.4 or greater before the activation block.

On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised

a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may

continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The unlockatblock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"unlockatblock": <Unlock
block height>}'
```

- **Time Delay:**
  The setunlockdelay parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"setunlockdelay":
<Unlock block delay>}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals
```

```
OPTION_TOKEN = 32                       // if set, this is a token, not a
native currency




OPTION_SINGLECURRENCY = 64              // for PBaaS chains or gateways to
potentially restrict to single currency




OPTION_GATEWAY = 128                    // if set, this routes external
currencies




OPTION_PBAAS = 256                      // this is a PBaaS chain definition




OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices




OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain




OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

```
}
```

The bridge definition has overridable defaults

```
  {
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/b1cfb315844850fdd2c89473fb7439c0c021e92d13e0a1db560303f0f83b0235/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/e90b011427be1aa5d78ac2e2d69bc96ee8c2fd8209a97cb51e19a395d0ced94f/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/be9785dfb3438de8a07c405dbfb71eca0477f45c875046a1bc561e819097d347/detection
Windows:
https://www.virustotal.com/gui/file/cab29d9bd73d9b1f1cd5dc86b6b6f3f13807d101ef4ae3f224d90e6788ede3b0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AVXjJgABQR9Xa2h4vsscUcQfw7A3tjs0Bk9F7thidKhl4cjgWL5Y9W6aWeuo4ljNYYUJ6NaNo8FXC8v
gP4cyNZI4ph/vMGlM" "/Downloads/Verus-CLI-Linux-v1.0.4-x86_64/Verus-CLI-Linux-
v1.0.4-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AVXjJgABQR9Xa2h4vsscUcQfw7A3tjs0Bk9F7thidKhl4cjgWL5Y9W6aWeuo4ljNYYUJ6NaNo8FXC8v
gP4cyNZI4ph/vMGlM"
56fbb10a12eb44d7dc139ff2cc3fa6a6a231976676257a2328a2856e3421baec
```

The result will be true if the signature is valid.

```
true
```

V1.0.3

Announcing v1.0.3 - MANDATORY MAINNET UPGRADE - HOPEFULLY THE LAST IN A WHILE - ACTIVATION DATE HAS BEEN PUSHED BY 2 DAYS

FOR CONTINUED MAINNET USE, UPGRADE TO v1.0.3 OR LATER BEFORE MAY 23, 2023, EXPECTED BLOCK #2549420

FOR CONTINUED TESTNET USE, UPGRADE ASAP

Mainnet: fixes:

- Fixes an issue discovered that could have resulted in behavior different than what has been tested in an edge case when mainnet activates

Testnet fixes/improvements:

- Forgives issues pre-1.0.2 on testnet that were blocking challenge resolution by lacking complete cryptographic proof for cross-chain notarization confirmation

What’s New

v1.0.3 has extended the API fundrawtransaction to allow Verus Mobile and other applications to easily construct transactions for DeFi conversions and cross-chain operations as well.

We expect at least one more non-mandatory release before the mainnet activation, Tuesday at block #2549420, which will add some pool merge mining improvements. Please make sure you update to v1.0.3 or greater before the activation block.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can

now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The unlockatblock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"unlockatblock": <Unlock
block height>}'
```

- **Time Delay:**
  The setunlockdelay parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"setunlockdelay":
<Unlock block delay>}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128                   // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

  "name": "PBaaSChain",
```

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

```
    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",
```

```
      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults


```
  {
```

```
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
  ],
  "initialsupply": 3000000
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/90e3d5d733afb7126ada94beb1e880f28af150846a5d64053d54ebc068b53cf1/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/43e52bed0f2aeda9570bef3a238fa2aab2018c0534f01bd449930f16d68b347d/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/73524e6469755c5727852ca423e1b52ed89e78f774d409e68de4d39e83166e2a/detection
Windows:
https://www.virustotal.com/gui/file/fe91ec477a3cfbdfd1ba2d05ce8dad48fec5b7e8e188d8f3433d87cd42f14d3a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AebXJgABQR877X9cSmsrmj3SE08+rgCb0pvwhp7JmSh+m0r2Swg9dhC4Yyra+GWBj0A0fcPH3o7OjvB
6P/BMfDsvxFeQVO7e" "/Downloads/Verus-CLI-Linux-v1.0.3-x86_64/Verus-CLI-Linux-
v1.0.3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AebXJgABQR877X9cSmsrmj3SE08+rgCb0pvwhp7JmSh+m0r2Swg9dhC4Yyra+GWBj0A0fcPH3o7OjvB
6P/BMfDsvxFeQVO7e"
b5ddb0576ac4d1ce5d6cd3c53946b8ca89bdc4e76f680caa822b181d8660199f
```

The result will be true if the signature is valid.

```
true
```

V1.0.2

Announcing v1.0.2 - MANDATORY MAINNET UPGRADE
UPGRADE TO v1.0.2 OR LATER IS MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

Mainnet: fixes:

- Fixes a regression that prevented staking on locked IDs as well as modifying locked IDs

Testnet fixes/improvements:

- Fixes issues discovered when challenges occurred on the Gravity chain that blocked resolution of cross-chain challenges.

- Addresses all known issues discovered in testing

What’s New

v1.0.2 has no feature changes, some minor API fixes, proof improvements, and we believe the third time's a charm ☺. The testnet will transition to the v1.0.2 protocol with no reset required, Wednesday, May 17, 2023 0:00:00, UTC.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an

unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts

for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The unlockatblock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"unlockatblock": <Unlock
block height>}'
```

- **Time Delay:**
  The setunlockdelay parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST setidentitytimelock "id@" '{"setunlockdelay":
<Unlock block delay>}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/47bcfc863a510ba1c2b73a90b4485bc586ab0a5b8865588a9dcda7a895bf1ce2/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/fb1f579d0f68e5bb00827a858af3665c9900939fccba89dcb9617739df3eedbd/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/95784a76519c6af601b83e346939a88487481a1eaa4ee0db67c307014631df69/detection
Windows:
https://www.virustotal.com/gui/file/53f06a9a5b64fd7a25f7830b013b69c95ea70ce11dbd54f1fb1058d8d5bc7c36/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AXm+JgABQR87e2irknAviTmXgC9Jfxda293fn6PtRZP2M0o2dMZ3JClcCC3Tjqy2ss+HwvBFrxvRBAI
6zvLOrkd6wwdr/fIz" "/Downloads/Verus-CLI-Linux-v1.0.2-x86_64/Verus-CLI-Linux-
v1.0.2-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AXm+JgABQR87e2irknAviTmXgC9Jfxda293fn6PtRZP2M0o2dMZ3JClcCC3Tjqy2ss+HwvBFrxvRBAI
6zvLOrkd6wwdr/fIz"
687ffc930582bbed1fffddb5aa406e580d3cb3e9eea890d6152a2c295d78e804
```

The result will be true if the signature is valid.

```
true
```

v1.0.1

Announcing v1.0.1 - MANDATORY Mainnet Upgrade, UPGRADE TO v1.0.1 OR LATER IS MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

Issues addressed

In addition to fixing an issue discovered by cautionfun#3236 on the current public testnet, this version also addresses an issue discovered during additional coverage testing in the oracle based, reversible rip-cord implementations to ensure the ability for the network to respond to any unexpected events quickly, safely, and without compromise to decentralization.

What’s New

In addition to the full PBaaS protocol and all capabilities previously described, version 1.0.1 will also include some new capabilities that were previously not in the PBaaS protocol. The capabilities basically extend the idea of a centralized currency to include the ability to launch temporarily centralized currencies, including those that can issue identities. This is achieved by setting the “endblock” on a centralized currency launch of either a token or liquidity basket, enabling:
1) Minting of currency on demand by the identity of the currency as part of the `sendcurrency`command.
2) Burning a liquidity basket currency, but instead of affecting the supply/reserve ratio as normal burns do, the identity of a centralized currency can also burn currency to affect the reserve ratio weights of the currency.
3) Registering identities on a currency. When an identity is registered on a centralized currency, the fee for the registration is not burned into the currency, but sent directly to the identity of the currency. This is true for token or liquidity basket currencies. For decentralized currencies, fees are burned, which on a liquidity basket, puts them into the liquidity pool for all LPs to share in earnings, and for a standard token, reduces the supply and available future registrations on that currency.
4) In v1.0.1, setting the endblock of a centralized currency causes all centralized capabilities of the same ID to end, converting it in whatever state it currently is in a decentralized currency with no

special control over it after that occurs by any identity on the network.
5) v1.0.0 did not allow a decentralized non-liquidity token to register identities. v1.0.1 does.

Thanks to ejuliano#8606 for suggesting the initial idea that led to the realization that these features could be added quite easily and with very little protocol risk. All of these new capabilities have had full coverage testing and validation over the past week thanks to @Asherda's leadership on that front.

Meanwhile, @alexenglish, @monkins1010, @asherda, quipacorn#5205, and others have been working on the Ethereum bridge deployment in preparation for this release. The testnet will transition to the v1.0.1 protocol with no reset required, Monday, May 8, 2023 4:00:00 PM, UTC.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-

knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition,

provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals
```

```
OPTION_TOKEN = 32                       // if set, this is a token, not a
native currency




OPTION_SINGLECURRENCY = 64              // for PBaaS chains or gateways to
potentially restrict to single currency




OPTION_GATEWAY = 128                    // if set, this routes external
currencies




OPTION_PBAAS = 256                      // this is a PBaaS chain definition




OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices




OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain




OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

```
}
```

The bridge definition has overridable defaults

```
  {
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/4d87f5f96a475d138a5e111ca7f13ddec4da546ccceb7308b742db08ecf8fb5e/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/6269809d114d14fe133bab31f945c29407f755627bb1787655ff8973ba67b096/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/eed44b0a8110269eb5b0d3db09a999be3e1f17bd91abe11c18a7ad20c1b47708/detection
Windows:
https://www.virustotal.com/gui/file/8e6e96a978c3690183654b497cf7d080d72e6ecb9e0070d2283e7b69c937777a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Ad+PJgABQR9Zgmo5lAenuOeur65N/uzpvHTnirNh7zZ5bp2jGAfDVWT0FlZEo3TjklPzhSG8rYq1yQL
Q/lCIsQaWcIF19TBB" "/Downloads/Verus-CLI-Linux-v1.0.1-x86_64/Verus-CLI-Linux-
v1.0.1-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Ad+PJgABQR9Zgmo5lAenuOeur65N/uzpvHTnirNh7zZ5bp2jGAfDVWT0FlZEo3TjklPzhSG8rYq1yQL
Q/lCIsQaWcIF19TBB"
7ebad8a3636a06159fd33a91c25c98538f01afcf44824a9c85fccfadd61b0c4a
```

The result will be true if the signature is valid.

```
true
```

```Announcing v1.0.1 - MANDATORY Mainnet Upgrade, UPGRADE TO v1.0.1 OR LATER IS
MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED
BLOCK #2546600
```

Issues addressed

In addition to fixing an issue discovered by cautionfun#3236 on the current public testnet, this version also addresses an issue discovered during additional coverage testing in the oracle based, reversible rip-cord implementations to ensure the ability for the network to respond to any unexpected events quickly, safely, and without compromise to decentralization.

What’s New

In addition to the full PBaaS protocol and all capabilities previously described, version 1.0.1 will also include some new capabilities that were previously not in the PBaaS protocol. The capabilities basically extend the idea of a centralized currency to include the ability to launch temporarily centralized currencies, including those that can issue identities. This is achieved by setting the “endblock” on a centralized currency launch of either a token or liquidity basket, enabling:

1) Minting of currency on demand by the identity of the currency as part of the `sendcurrency`command.

2) Burning a liquidity basket currency, but instead of affecting the supply/reserve ratio as normal burns do, the identity of a centralized currency can also burn currency to affect the reserve ratio weights of the currency.

3) Registering identities on a currency. When an identity is registered on a centralized currency, the fee for the registration is not burned into the currency, but sent directly to the identity of the currency. This is true for token or liquidity basket currencies. For decentralized currencies, fees are burned, which on a liquidity basket, puts them into the liquidity pool for all LPs to share in earnings, and for a standard token, reduces the supply and available future registrations on that currency.

4) In v1.0.1, setting the endblock of a centralized currency causes all centralized capabilities of the same ID to end, converting it in whatever state it currently is in a decentralized currency with no special control over it after that occurs by any identity on the network.

5) v1.0.0 did not allow a decentralized non-liquidity token to register identities. v1.0.1 does.

Thanks to ejuliano#8606 for suggesting the initial idea that led to the realization that these features could be added quite easily and with very little protocol risk. All of these new capabilities have had full coverage testing and validation over the past week thanks to @Asherda’s leadership on that front.

Meanwhile, @alexenglish, @monkins1010, @asherda, quipacorn#5205, and others have been working on the Ethereum bridge deployment in preparation for this release. The testnet will transition to the v1.0.1 protocol with no reset required, Monday, May 8, 2023 4:00:00 PM, UTC.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first

unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a

solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**

  The timelock parameter defines the unlock height of the identity.

verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0, "timelock":

- **Time Delay:**

  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2, "timelock":

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

verus -chain=VRSCTEST getcurrencyconverters btc eth

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10, "convertto":"VRSC-BTC"}]'

```
- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**
```

verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10, "convertto":"BTC","via":"VRSC-BTC"}]'

```
- **Preconverting to new currency, NEWCOIN, before it is active:**
```

verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10, "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'

```
- **Sending VRSCTEST cross-chain to PBaaSChain:**
```

verus -chain=VRSCTEST sendcurrency "*" '[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10, "exportto":"Bridge.PBaaSChain"}]'

```
- **Converting VRSCTEST cross-chain to PBaaSChain:**
```

verus -chain=VRSCTEST sendcurrency "*" '[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10, "convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaaSChain"}]'

```
- **Converting PBaaSChain to VRSCTEST:**
```

verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'

Defining a Currency

Currency Options

OPTION_FRACTIONAL = 1 // allows reserve conversion using base calculations when set
OPTION_ID_ISSUANCE = 2 // clear is permissionless, if set, IDs may only be created by controlling ID
OPTION_ID_STAKING = 4 // all IDs on chain stake equally, rather than value-based staking
OPTION_ID_REFERRALS = 8 // if set, this chain supports referrals
OPTION_ID_REFERRALREQUIRED = 16 // if set, this chain requires referrals
OPTION_TOKEN = 32 // if set, this is a token, not a native currency
OPTION_SINGLECURRENCY = 64 // for PBaaS chains or gateways to potentially restrict to single currency
OPTION_GATEWAY = 128 // if set, this routes external currencies
OPTION_PBAAS = 256 // this is a PBaaS chain definition
OPTION_GATEWAY_CONVERTER = 512 // this means that for a specific PBaaS gateway, this is the default converter and will publish prices
OPTION_GATEWAY_NAMECONTROLLER = 1024 // when not set on a gateway, top level ID and currency registration happen on launch chain
OPTION_NFT_TOKEN = 2048 // single satoshi NFT token, tokenizes control over the root ID

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.

So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.

First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency '{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":[0.01],"minpreconversion":[1000],"preallocations":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.

all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation

in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.

After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.

Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":[10000,5.1298]},
"initialsupply":20000'

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'

Defining a PBaaS blockchain

```json

  {
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/4d87f5f96a475d138a5e111ca7f13ddec4da546ccceb7308b742db08ecf8fb5e/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/6269809d114d14fe133bab31f945c29407f755627bb1787655ff8973ba67b096/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/eed44b0a8110269eb5b0d3db09a999be3e1f17bd91abe11c18a7ad20c1b47708/detection
Windows:
https://www.virustotal.com/gui/file/8e6e96a978c3690183654b497cf7d080d72e6ecb9e0070d2283e7b69c937777a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Ad+PJgABQR9Zgmo5lAenuOeur65N/uzpvHTnirNh7zZ5bp2jGAfDVWT0FlZEo3TjklPzhSG8rYq1yQL
Q/lCIsQaWcIF19TBB" "/Downloads/Verus-CLI-Linux-v1.0.1-x86_64/Verus-CLI-Linux-
v1.0.1-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Ad+PJgABQR9Zgmo5lAenuOeur65N/uzpvHTnirNh7zZ5bp2jGAfDVWT0FlZEo3TjklPzhSG8rYq1yQL
Q/lCIsQaWcIF19TBB"
7ebad8a3636a06159fd33a91c25c98538f01afcf44824a9c85fccfadd61b0c4a
```

The result will be true if the signature is valid.

```
true
```

Announcing v1.0.0 - PBaaS Mainnet Upgrade, UPGRADE TO v1.0.0-x MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

What’s New

This release delivers on every part of the vision (and most stretch goals) either described in the vision paper of June 2018 or ever planned for the PBaaS release. Thanks to the incredible combined efforts of so many people in the community ranging from development, companies and projects joining and their open contributions (Valu/Arkeytyp, CHIPS, vDEX, VaultAlert, cragslist, and more) to community members supporting users to helping educate others who can contribute as well, we have run these protocols now for years as we’ve improved them, and all of these capabilities will go live on the Verus mainnet May 21, 2023, with the activation expected targeting block 2546600.

As a community, we’ve discussed what the Verus PBaaS protocol can do for years. Now that it has existed on testnet for as long as it has, only those who have either been part of those conversations or experienced it first hand have a real sense for how much better Verus PBaaS is as a solution for cross-chain, DeFi AMMs, decentralized markets, self-sovereign provable recoverable IDs, NFT capabilities, anti-phishing, anti-MEV, scale, or just about any of the challenges people have on crypto platforms today. In fact, as we prepare for activation of this protocol that enables so many new use cases for crypto, not to mention easy onboarding when entrepreneurs discover how to really leverage VerusIDs, PBaaS is so far beyond what people are experiencing on any crypto platform today that it has been easy to dismiss our community as describing the impossible.

Once we are live on mainnet and people can actually use all of this themselves, along with every project, chain and currency on the PBaaS network and EVERY ERC20 or ERC721 on Ethereum or even bridged to Ethereum, the truth of Verus will be self-evident, and those who build on Verus will more easily build faster, better, more secure apps and services with the possibility of provable identity + privacy, crowdfunded projects, businesses, economies, and public infrastructure efforts all seamlessly integrated into UIs that do not need business deals, permission, or any centralized infrastructure to connect services and users, enabling everyone to communicate in provable, private, or public ways that always include bidirectional, secure commerce of all kinds.

This core capabilities of this release include (lots to learn to understand it all, but here’s a partial list):

1) **Mainnet PBaaS Activation MAY 21, 2023, EXPECTED BLOCK #2546600**: v1.0.0 is MANDATORY, meaning that you will need to upgrade to this release or a later version v1.0.0-x, before MAY 21, 2023 to stay reliably connected to mainnet. Testnet will also be reset within the next day, and this release will allow you to connect to the new testnet, which we see as a core development platform and staging area for the Verus PBaaS network and have no plans to reset in any foreseeable future.

2) **Verus Intersystem Protocol (VIP) - Layer 0 Multichain and Inter-chain Protocol**: To our knowledge, the VIP protocol is the only fully decentralized, provable cross-chain technology available on any network that is based on cryptographic proofs of each chain with optional witnesses to confirm chain state. From what we've been able to learn about LayerZero, Cosmos, Polkadot, Thorchain or others, VIP is different from (though closest in some ways to LayerZero), superior to, and more decentralized than all other cross-chain protocols we have seen. We can of course get into any level of discussion on the topic, but expect more descriptions and educational material to follow, now that the protocol was originally conceived of 5 years ago, and has been developed and tested/hardened for years.

3) **VerusID Content Multimaps - Unlimited Provable Data for Every VerusID**: With the PBaaS upgrade, VerusIDs gain a powerful new capability that can be used for social networking, voting, oracles, publishing any information, various forms of provable attestations, ratings, and much, much more. With the scalability of PBaaS and VerusID content multimaps together, the Verus PBaaS network becomes the most scalable, permissionless, provable source of human data from which AIs can learn about humanity in the world. We don't believe anything else even comes close, and with the recent rise of AI, it feels great to have this come together as a core original goal at a very appropriate and important time in history.

4) **Permissionless Chain, Token, and Liquidity Basket Currency Launches**: Whether you are looking to crowdfund an effort in a manner much like Kickstarter or IndieGoGo, sell identities to which you expect to add value or that are sub-IDs of a very cool root ID, or launch an entirely new blockchain economy and blockchain that starts with all of the Verus technology and a bridge to Verus and Ethereum from day 1, Verus is probably the best platform for you. In fact, as we were planning for the next testnet and wanted to have our own version of DAI to use in the bridge, since we want only decentralized currencies in the live mainnet bridge, we used a command to launch a token on the old testnet, exported it and all its supply to Ethereum's Goerli on the existing bridge, and that will be the DAI proxy/simulant that we will use on the new testnet. It saved us all some time, proving that Verus PBaaS will also be the easiest, most efficient way to launch even an Ethereum ERC20, whether that ERC20 represents a token, DeFi basket, or even another PBaaS blockchain ☺

5) **Decentralized Ethereum Bridge**: Shortly after PBaaS activates on mainnet, the Ethereum bridge will launch as a decentralized gateway and 1:1 provably mapped currency called “vETH” on the Verus network, also available to all PBaaS chains. The Ethereum bridge will also include a 100% backed basket of 33% Verus, 33% Ethereum, and 33% DAI, which will have the following functions:
Auto conversion of fees from Verus <-> Ethereum when sending cross chain, based on the on-chain conversion price in the liquidity basket
Permissionless ability to register *.vETH IDs, which in addition to the normal sub-ID capability of creating a single token that has control over the ID and can be exported to Ethereum as an automatic ERC721 NFT, can alternately be used to create a “mapped currency”, which can be provably mapped 1:1 to any ERC20 currency on the Ethereum network by exporting it to the vETH bridge. Only Verus root IDs, and IDs of a gateway can create mapped currencies to a gateway.
Decentralized, fair bridge launch. The vETH bridge will launch as other currencies, chains, and gateways can launch, including the bridge converter. When the bridge launches, the contracts will have a surplus of Verus that comes from the fees paid to launch the gateway. In the protocol, that surplus is first used to solve the chicken and egg problem of none of one currency on the other chain. The contracts will cover the Verus fees for all sends from Ethereum before the Bridge converter launches, the liquidity basket becomes active, and the bridge recognizes that launch. Once that happens, the remainder of ½ the fees (5000 VRSC) that are left over from the launch and did not go to miners and stakers or pay Verus fees for people sending from Ethereum

6) **The Verus Fee Pool and Rewards**: Another technology and solution unique to Verus, the Verus Fee Pool technology goes live with PBaaS. What this means is that as people use the cross-chain, DeFi, or purchase/register IDs on the network, the miner or staker will put all fees into the on-chain “fee pool”, and then take 1/100th of the pool in addition to the block reward. This gives everyone incentive to still prioritize transactions based on fees, while preventing validators from gaming the fee system by washing fees-in vs. fees-out or working to reorg/rewind to capture a particularly juicy block reward. Instead, this technology aligns all validator incentives with the health/proper operation of the network and creates a circular on-chain economy that can last well beyond block rewards.

For those who actually understand the challenges actual users have in crypto today, this release represents a historic move forward in public cryptographic networks and credibly and actually neutral infrastructure for society-wide human and AI collaboration and learning. As a community, this version really does realize the vision laid out in the original Verus Vision Paper, and at the same time provides much more than was described there. We are a community of individuals, and it is only because that is what we are that we have all arrived here together. The rest of the world is stuck on their Munchausen’s protocols, trying to figure out how to share the front-running and back-running spoils taken from users, but Verus activation gives them a better path forward.

As always, we do not promise, we describe a vision and welcome contributors from everywhere. Now that we, as a community, have delivered on the original phases of the vision together, it is time for all of us to consider what we might want to build over this incredible, geoscale, ID-enabled, fully decentralized platform. Our next efforts across the community should be to realize the promise of this network in the use cases we create, use cases that we will have an advantage creating on Verus over any other platform because it really is that much better.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/9700f0d571c9ec1a399b424d862d601ee0a54ee3475c5a851474d3016655263c/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/a7a18bbf80c119fa027bb1df0ae423ce934b6e7448f325f1a51c8b74e968c906/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/2f1f7927e52b5a29c448021e3354a49add03bc87203726c482d75c0b90e5cf6b/detection
Windows:
https://www.virustotal.com/gui/file/af996552413cc186ca482e472b980a55efa21a7819e1ed2bbe5392a41f0b971a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AY8+JgABQSBR3wd5yPxBDCKh5vnf6hc3EGmgvR9ZcfIHIlw9P1V5eCNG9jxkOMFx3Gh9s2DUIM1HDdJ
w7UGlNAPYWJmcnA8/" "/Downloads/Verus-CLI-Linux-v1.0.0-x86_64/Verus-CLI-Linux-
v1.0.0-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AY8+JgABQSBR3wd5yPxBDCKh5vnf6hc3EGmgvR9ZcfIHIlw9P1V5eCNG9jxkOMFx3Gh9s2DUIM1HDdJ
w7UGlNAPYWJmcnA8/"
2bd9b9387ad58ebc208cfa83513fb8b5ba3226401fe8fe1fcdba607e780cf869
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9-5 - CRITICAL PRIVACY FIX PARSING FRIENDLY PRIVATE NAMES, CRITICAL UPDATE FOR RELIABLE TESTNET OPERATION

Mainnet Changes - Wallet Hardening

v0.9.9-5 Fixes an issue introduced in v0.9.9 during the ID name hardening work that prevented parsing id@:private to the z-address of id@ when running a native node. Instead, such an address parsed to the main id@. Since this issue could impact on-chain privacy of someone intending to send to or from a private address and using friendly names, we are considering this a critical privacy release.

This version also prevents the wallet from crediting transactions that can be constructed to make a burn look like a spendable output, ensuring that unspendable funds are not added to a wallet or address balance. @Alrighttt, who has been focusing on Verus security testing, reported a way that someone might create such an output, and we appreciate that it is before we have ever seen such transactions on the public or test networks. Thanks @Alrighttt for the report and discretion that helps keep them off the network(s) altogether! We recommend that everyone update as soon as possible to version 0.9.9-5 or later.

We are working towards the mainnet PBaaS upgrade version 1.0.0, but we cannot put a specific date beyond SOON on its release.

What’s New for TestNet

This release fixes hopefully the last remaining issue that was causing testnet users to experience sync issues on PBaaS chains.

There are no other significant changes.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/c72b18c9ad304874f47502d240b494b1b82d9d82c4953769804ccf4e434de148/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/350d646df8d6b3ec204695852cd415f1ef65e4177475cee4dbe62daab3f04906/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/177d6a8a4e2693ddd0c144f47f6dac477aedaac5a02d0b7b05e2914bb67bf43f/detection
Windows:
https://www.virustotal.com/gui/file/7d2fbe3b1d79988fa65a57d8310666d37fd62ac0eb3321462a8ebb249a47bf00/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Adu/JQABQSB8RI2SQ6GBoJeUrJB1yficCCbEJ8o9oMgjS34zTJv0KD8hMX1urP+vzjMmmtBpsf/
Ax1UXW+1Mz/1n665VCrXc" "/Downloads/Verus-CLI-Linux-v0.9.9-5-x86_64/Verus-CLI-
Linux-v0.9.9-5-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Adu/JQABQSB8RI2SQ6GBoJeUrJB1yficCCbEJ8o9oMgjS34zTJv0KD8hMX1urP+vzjMmmtBpsf/
Ax1UXW+1Mz/1n665VCrXc"
9eb4bbfc466e2a8e63cae3b90490ea6779e894ab4381256418516df96f93fab2
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9-4 - RECOMMENDED FOR MAINNET, MANDATORY UPDATE FOR RELIABLE TESTNET OPERATION

Mainnet Changes

Low level stakeguard improvements, no functional changes

What’s New for TestNet

This release fixes an issue that was causing testnet users to experience sync issues on chains that had more than one oracle upgrade for the same capability activated.

There are no other significant changes.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can

now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/d4e1e6fe568145b847f8a8b46fd21bb751a83bd9cd9849aae68a590e08c16d0e/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/e2730182906fd866728c23cfd5edcaa7f33a4941a730ae6d06cd193b04e9a924/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/1e4bb2641771d4121e235b5937947d8e0e28d929ed8aa79340390b2ffae11b4d/detection
Windows:
https://www.virustotal.com/gui/file/afb3fee31caeab77cfdbb6a8765e80ae418c25ce61002baea73f6fc3a2f75bb5/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AUqqJQABQR9eyv2N6JbDY8HAvGMn5mR2NYLW3sA0jbud0we0KNyGuAV4Upp/qPgl9265PL4ZkjBLY45
BrkIstubavegqwn+u" "/Downloads/Verus-CLI-Linux-v0.9.9-4-x86_64/Verus-CLI-Linux-
v0.9.9-4-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AUqqJQABQR9eyv2N6JbDY8HAvGMn5mR2NYLW3sA0jbud0we0KNyGuAV4Upp/qPgl9265PL4ZkjBLY45
BrkIstubavegqwn+u"
f5577a74e2d279b62aa42d85359f8d608ebc1ad7464556251ea29e7a36b6472c
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9-3 - FULLY OPTIONAL UPDATE FOR MAINNET, MANDATORY UPDATE FOR CONTINUED TESTNET OPERATION

Mainnet Changes - no protocol changes

V0.9.9-3 introduces the `prunespentwallettransactions` CLI command and RPC API, providing a method for holders on testnet or mainnet of very large wallets to prune older spent

transactions, which can improve performance for wallets that may be slowing down due to size. Thanks @Himu007 for the PR! There are no mainnet protocol changes in v0.9.9-3 over the prior release.

What’s New for TestNet

This release includes an activation that allows a major upgrade to the testnet Ethereum bridge contracts that is both not expected to be needed again on testnet and not ever needed on mainnet with the new contract upgrade model. This upgrade should clear all known issues on the bridge to Goerli, meaning any remaining bridge transactions will be delivered, and the ETH bridge should then be fully functional.

The Path for PBaaS to Mainnet

This testnet has actually been through a lot already, more than we expected before mainnet, and this last experience, along with its full restoration of cross-chain and all function has been very useful and was the impetus for the Verus Oracle Notification technology. In the process of getting here, we implemented the necessary changes and fixes to get everything back on track, but we also have remnants of these events in the daemon, and on the final protocol, these are there to enable compatibility with the history of testnet, not because we need these upgrades on mainnet PBaaS. It gave us experience, increased confidence, and pushed us to develop a fast, decentralized network response capability. At the same time, it would be best for our code base, both on the Verus side and in the ETH contracts if that testnet history is removed from the code before mainnet activation. Doing so would make the next upgrade incompatible with today’s testnet and its earlier history.

As a result, we have decided that we will create one final testnet, but this time, we would do it a little differently than in the past. We will extend the pre-PBaaS upgrade on the existing testnet to a future date via the oracles. If you have started a chain and control the default oracle on any chain, please check the #pbaas-development channel for instructions on setting your oracle to extend the upgrade.

We will leave the current testnet running and fully functional, as we prepare the next release, which we expect to be ready for a mainnet PBaaS activation. We will effectively abandon this testnet and leave it running long enough to overlap with the next and final testnet instance before PBaaS release. You should be able to actually send currencies over from the first testnet to the second via Ethereum's Goerli testnet, even map some VRSCTEST-CLASSIC or something else through the bridge, but we do not intend to update that daemon again, or continue to run nodes or notaries on the old test network for more than a couple weeks after the new testnet goes live.

Thanks for all your help and contributions, both already made and those to come for this historic upcoming release!

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents,

marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/d59a5a175ac71d43cf4c69076de70fb0a97a5cd09569c3beedd1bddd1255c872/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/0ad8752a41a29b469fa2d62d493ec088a65c7b2f514a2035fcc0e64b2ea05010/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/9d1fe67ec99f3da3d300c3294af4ad3458015f1a6b1552b7b6be83b07909626b/detection
Windows:
https://www.virustotal.com/gui/file/e243faa448e5f941d7fffe85fb4df12f461ed525d33521b6d774f9e9144066cd/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AcKVJQABQR8REvfAnC1o2X3i+ya+XijCwxWtNunasCWLtCP5tZsdDm9Zg0kxb8k95KWLZwH7QZMkirX
reo/bNHmo+c5DeKgE" "/Downloads/Verus-CLI-Linux-v0.9.9-3-x86_64/Verus-CLI-Linux-
v0.9.9-3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AcKVJQABQR8REvfAnC1o2X3i+ya+XijCwxWtNunasCWLtCP5tZsdDm9Zg0kxb8k95KWLZwH7QZMkirX
reo/bNHmo+c5DeKgE"
50229f3f126cc77abaf6ea4e28d27d6c5ba25acbbaae9c6bff48b1f73924ead0
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9-2 - CRITICAL SECURITY UPDATE FOR MAINNET AND REQUIRED UPDATE TO RESUME SUCCESSFUL MERGE MINING NOTARIZATION ON TESTNET

Important Mainnet Security Updates**

v0.9.9-2 introduces new mitigation code for the **Rab13s** vulnerability, announced by Halbourne 3 days ago as a zero day vulnerability in over 280 blockchains, including Zcash and Verus, that could allow someone to attack and bring down specific nodes that they may target through exhausting the node's resources. The Verus fix adapts a fix implemented in Zcash that also does a better job of preserving privacy while gossiping transactions around the network by making it harder to track their node of origin via listening nodes.

This release also includes an important update of the OpenSSL library, which Verus uses for encrypted node connections and addresses a buffer overflow vulnerability in that library.

What's New for TestNet

This release re-enables cross-chain notarization, and represents the completion of a great deal of learning from the first roll out of auto notarization. Basically, we encountered two issues:

1) There was a notarization bug in a significantly updated approach that caused notarizations to fork, effectively simulating a worst case, unmitigated attack on the notarization system. While that did not cause the system to fail per se, it resulted in the generation of many perceived, although not real, notarization forks on the network.

2) Due to our effort to err on the side of more evidence than we believed was sufficient with auto notarization, then optimize to reduce the proofs required, the protocol was simply taking too long and generating too large proofs to be reasonable, required, or what we need on mainnet in any real scenario.

As a result, we will take a few days to change some parameters in the auto notarization protocol that will both reduce the total and maximum amount of proof required, as well as provide a sufficient level of cryptographic proof, similar in strength to the current proof model with more efficient and selective proof choices. We don’t expect this important step to take long, and until then, we will leave testnet on the pre-PBaaS to mainnet preparatory state.

The good news is that since we have the new Verus Notification Oracle technology, we can enable everyone running on the current testnet to remain connected as all cross-chain transactions to and from PBaaS chains will resume flowing. We are still seeing an error in BridgeKeeper, but since this release includes a mainnet security update and addresses all other known issues on testnet, we’ll release this immediately and update Bridgekeeper as soon as possible. In order to merge mine PBaaS chains with Verus or the Ethereum bridge to Goerli going forward, you will need to update to v0.9.9-2, which everyone should run for mainnet anyhow.

Once you update, please feel free to resume merge mining all PBaaS chains and Verus. You will need to wait for the BridgeKeeper update to be able to merge mine and stake with Ethereum’s Goerli.

Thanks for all your testing! We’ve learned a great deal from this upgrade, and although it may feel like a delay that we took the time to learn and address it in place on the public network before moving on, this learning and consequent hardening will ensure the mainnet release will be more hardened than ever. Next steps are to upgrade auto notarization to be resilient to anything like this level of buildup over weeks automatically and prepare the mainnet release. Join us in testing everything in the #pbaas-development channel. Contribute as you learn and build, and let's all make history together!

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@"

],

"minnotariesconfirm": 2,

"nodes": [

{

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/0e5e685be2d8a28b805ecc85cfff52114aafb593343c1221390976543e03a1de/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/adc5382cf84b84955a0601b33e634e4c3e15b9c543d1224a87785815f357646e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/b94081663983ab4412eea7939d9ef1a5618767e586a48693dccd825a2a619d7c/detection
Windows:
https://www.virustotal.com/gui/file/4a4dde2751f3deae34d4795daae5a981a7541a568f59d83eece5cb895cf0ef34/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Ae1xJQABQSCusp3AYTcIlTUklLlDl7goo3n1773y1jIR73JtLLHFgjjwpC/tvuDcJ8hdj/
voxmHU2WmTKq7LkVaTAi/EZD1V" "/Downloads/Verus-CLI-Linux-v0.9.9-2-x86_64/Verus-
CLI-Linux-v0.9.9-2-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Ae1xJQABQSCusp3AYTcIlTUklLlDl7goo3n1773y1jIR73JtLLHFgjjwpC/tvuDcJ8hdj/
voxmHU2WmTKq7LkVaTAi/EZD1V"
51813c20a4fd94870987dec24bc41606f27bd26c29952ea12f51a8204a39ae37
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9-1, MANDATORY FOR TESTNET, no mainnet changes

This release implements a fix for communication with BridgeKeeper for the Testnet Goerli Ethereum Bridge, as well as edge-case notarization improvements. We need every testnet participant to upgrade to this version as soon as possible to resume notarizations for the ETH bridge and the pbaas chains. Once enough people upgrade, the testnet upgrade can be broadcasted via oracle and we can get closer to getting pbaas on mainnet.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

```
    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {
```

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/0eae0bb549e611d24315e726e4f10a13f4c9c548c9158cb3988e036b279da4c3/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/5455afbf15a4c0caace25ffab4aa1ebd163db9ff4eec929dbdbf53a2c152535d/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/f4ce5271b9c35cba144b8f79f2552590ce805fd8265d20aa156c4bb6d04ad29e/detection
Windows:
https://www.virustotal.com/gui/file/faa39e0b61fab0dda9e697deafc3d3250a81c737349db7ea4f64583969135daf/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AVBfJQABQSAcInXeVZTA3kTri4vGTRQjZwceXzhrMY6bMy+vLatPSjONfaXXrDU6N2sHsiYqNuHRs+v
zm9brZDmk1NlqNdNl" "/Downloads/Verus-CLI-Linux-v0.9.9-1-x86_64/Verus-CLI-Linux-
v0.9.9-1-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AVBfJQABQSAcInXeVZTA3kTri4vGTRQjZwceXzhrMY6bMy+vLatPSjONfaXXrDU6N2sHsiYqNuHRs+v
zm9brZDmk1NlqNdNl"
3bda41e7f53ab13a0f75a45514974a20d7ca470c1ccb9fcbb9743c02c68e3642
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.9 - PBaaS Mainnet Preparation Release, MANDATORY FOR MAINNET AND CONTINUED TESTNET USE

What’s New

There is so much new and important in v0.9.9, and we are so close to the mainnet PBaaS release that this is a hard announcement to keep to a digestible size. Key points:

1. **Preparing for #LaunchPBaaS**: v0.9.9 is MANDATORY, meaning that you will need to upgrade to this release ASAP to stay reliably connected to mainnet after the first preparatory PBaaS upgrade is triggered, both for testnet and mainnet. While the current deadline for being upgraded is in 1 week, it is a soft fork that we would like to activate as soon as possible, and if we can believe that we have more than 50% of the validating network updated earlier than 1 week, we will trigger the activation with our Oracle Notification Technology described in 2. **(Mainnet + Testnet)**

2. **Enabling Seamless Decentralized Network Upgrades**: This release includes a Verus invention of consensus-capable oracle notifications built into the core protocol, which enable dynamic coordinated consensus changes that can be triggered at a time decided after the actual release which enables them, all without any compromise on decentralization. The Verus Community may decide that we are ready to go earlier than one week on this upgrade and activate it, even before its scheduled time. The good news is that this first test upgrade will primarily affect testnet and prepare for the mainnet PBaaS upgrade, so it is a perfect test case for a mainnet test run before we #LaunchPBaaS. This release actually has the ability to upgrade mainnet to PBaaS, but we believe it is prudent to have at least some time on this new protocol on testnet before the v1.0 release, which will upgrade mainnet. If we make no additional changes to the protocol in this release, which we do not intend to do at this time, this version may be able to follow the v1.0 upgrade. If we determine that it cannot, the oracle upgrade technology will enable the daemon to recognize that it is not capable and request that the user upgrade to a newer version without requiring a bootstrap for the upgrade. **(Mainnet + Testnet)**

3. **This upgrade fixes all known testnet issues with no reset**: Using v0.9.9, you should be able to sync fully and resume all cross-chain operations on the Verus test network, if you were not able to perform cross-chain operations or sync to the network on the prior version. As mentioned, someone seems to have mined with an intermediate version along the way on testnet and made the network unavailable without a bootstrap on the older daemon. This version should fix that issue as well as any cross-chain difficulties. Auto-notarization will be enabled on testnet in sync when the testfork executes. Until then, all witnessed cross-chain operations should be able to resume, if they have had difficulty. **(Testnet until the #LaunchPBaaS release)**

4. **Bridging Blockchain Networks Without Witnesses**: v0.9.9 is the first release of any blockchain protocol anywhere that we know of which enables cryptographically provable, cross-chain bridging across Verus and all PBaaS and other connected blockchains in the network, either combined with or independent of the need for notary witnesses. With notary witnesses, cross-chain transactions complete faster, but they are also checked against

cryptographic evidence, making it difficult to impossible for anyone, including notary witnesses to take any action, even colluding with validators that would go against the wishes of the most chain power. We have been referring to this new cross-chain consensus technology as “auto-notarization”, as that is what it actually is, but it would also be accurate to call it Proof of Proof of Power, as that is how it works. For purposes of ensuring everyone understands the state of the technology, it is fully functional and we have taken great care to ensure it is sufficiently secure, but we have not yet had the protocol proven to an academic standard. Because of that, while it provides a security fall-back against misbehaving notary witnesses and will be fully functional and we believe sufficiently secure without notary witnesses on release, we currently recommend toconsider the protocol unproven at this time and that any serious PBaaS chains be launched with operational notary witnesses until further notice.

Using `“notarizationprotocol”:1`, which is also the default, a chain launch can specify notary witnesses that will be used when they do their job. Even if they stop witnessing for any reason, the protocol will fail-over to auto-notarization, ensuring that cross-chain transactions still function as expected, even if just for people to move their assets to another blockchain, although significantly more slowly and based solely on cryptographic proof and challenge resolution between merge miners and stakers. This cross-chain proof and challenge protocol operates even when witnesses are also operating and if miners and stakers continue to prove a more powerful chain than the one witnesses represent, the most powerful chain can still be confirmed. **(Testnet until the #LaunchPBaaS release)**

5. **Expanding L1 VerusID APIs**: This release provides a major enhancement to the identity content multimap support with two new APIs, `getidentityhistory` and `getidentitycontent` that provide real-time mempool access and also introduce self-sovereign delete operations to identity content that can be used by applications, making it easy for applications to index, organize, and aggregate an unlimited amount of user content referenced by an identity and made available to the application. This is the technology we used for oracle upgrade notifications, as all of this is available at L1, making it possible to imagine future protocols that can deal smoothly with diverse community opinions worldwide, but always respect the sovereignty and decisions of all members on the network. **(Testnet until the #LaunchPBaaS release)**

6. **Introducing Restricted VerusID Staking for PBaaS Chains**: This release enables the stretch goal of restricted ID staking as an option for launching PBaaS chains. This option means that on such a PBaaS chain, only funds controlled by IDs with a parent of that PBaaS chain may stake blocks on that chain. When combined with restricted forms of ID issuance,

whether approval or referral required and/or referral rewards, this capability opens up an entire new area for public/private blockchains and applications, as well as government, voting security, corporate, or organizational use cases.**(Testnet until the #LaunchPBaaS release)**

Additional Verus Capabilities

7. On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

8. On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

9. On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/7816792bbf33f937a9ea0ef6c651005a5f9133ac60dcd3b117f9fc2d44a9a6c3/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/c4b848f380add524da457fe63e97ddd3682a156384b5d0a9605c1469fb3f66fb/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/20b6ae2af76015eafa7436bf8531f857e4d749cc2d976287bcceeff3b88a7feb/detection
Windows:
https://www.virustotal.com/gui/file/d75ad92d552d660719785ec53de6f9c766e6bfdd476b2b8720d001af1bd6d12a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AbxTJQABQR+xQsQKWDOIJclDMUHezwk6ghlNOrmDZn5YWoYF4tLodgmAdZu9pefSBP4iCjAm5fFG/
6qD44bP0vv1dvdEsz6e" "/Downloads/Verus-CLI-Linux-v0.9.9-x86_64/Verus-CLI-Linux-
v0.9.9-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AbxTJQABQR+xQsQKWDOIJclDMUHezwk6ghlNOrmDZn5YWoYF4tLodgmAdZu9pefSBP4iCjAm5fFG/
6qD44bP0vv1dvdEsz6e"
dc0b5adb8bf5f3375e1be7ff10dc35ba34d7af31269c5254157f5ecf0e12635b
```

The result will be true if the signature is valid.

```
true
```

v0.9.6-2, CRITICAL UPDATE FOR MAINNET
This release contains a critical update for the Verus daemon. This version does not support testnet, but a future version will be released shortly to support it. For more information on running testnet, please refer to the previous release or the next one.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents,

marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/3ba7722ead9c8da84acad5a381cf9632a2f76fa4f170f9a5721d38919ae51e40/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/4c46c71daadfb9b0d66b0c5bc5e6b1ab2a61276dec2153ccb7e9cc096ad35e8b/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/f5eb6d1cbdde1fad413956e563ff0c7eba1f3f2aa8fb60f99d6e7ad24072d185/detection
Windows:
https://www.virustotal.com/gui/file/1afcb9d344271cc3bbd2946c7adaaaac25848567eda7dceecdb309a68c3792bf/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AX76JAABQSD/CpNTPN3K3j8wFcIMRYCMh8MDeIwAMK+QAFnlnQoWB2UIM3g7D97KlPAk9T7RPBoXRsU
kDK9wtgPDw1XdWH6O" "/Downloads/Verus-CLI-Linux-v0.9.6-2-x86_64/Verus-CLI-Linux-
v0.9.6-2-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AX76JAABQSD/CpNTPN3K3j8wFcIMRYCMh8MDeIwAMK+QAFnlnQoWB2UIM3g7D97KlPAk9T7RPBoXRsU
kDK9wtgPDw1XdWH6O"
f7764f3299120d97a250547287d1beac9a44f4003c433072fe7c9c6cadf28630
```

The result will be true if the signature is valid.

```
true
```

v0.9.6-1, RECOMMENDED UPDATE FOR MAINNET, CRITICAL FOR CONTINUED TESTNET AND ETHEREUM BRIDGE (GOERLI) USE

What's New
THERE IS NO TESTNET RESET, AND ALL OTHER NOTES REMAIN THE SAME AS LAST RELEASE.

v0.9.6-1 has no mainnet changes and fixes the following testnet issues:
- Fault when initiating mining and staking on a small number of machines
- Sync from scratch issues on testnet

This release fixes all reported issues in all community testing. There are no protocol changes, and no known changes required. All network functions should work as intended.

Verus PBaaS and DeFi RC6 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don't expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (3 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security

hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

If you have reported a bug, since the bounties for the 0.9.5+ versions were announced, you are on the list, and someone from the Foundation will reach out to you in the near future.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release this coming week, and start building your vision today!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

```
    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {
```

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/3f1b2713bd7ba9ab01af59834a90720c734e773a9f73cd095a79248cc5a72b28/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/4ef4dbc1f81acb454ba27427e05bbee552b519de3564c4485b36349966c70c81/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/cee56134a3ae6ad6319405b8e058aa0ca650bfccb69436a84888a50fd5892abb/detection
Windows:
https://www.virustotal.com/gui/file/5a2e3252f67198cc7b4a247621dd1c8d3d5ecf1d218b2142fa08f8b52cd6c611/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Aa2MIwABQR+fAcDKYxskjVHMQGGDUkWI4NhJdUS/cotKm8S2OucQhk0/p9FDc9OzvHtyU5FMtuhExly
Nabb4+ona6dyu21uu" "/Downloads/Verus-CLI-Linux-v0.9.6-1-x86_64/Verus-CLI-Linux-
v0.9.6-1-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Aa2MIwABQR+fAcDKYxskjVHMQGGDUkWI4NhJdUS/cotKm8S2OucQhk0/p9FDc9OzvHtyU5FMtuhExly
Nabb4+ona6dyu21uu"
8cf1ce007e9bd2ff6d1ee68c501e82b5a22ec6eaedab1e5a3c27aa2221bd414a
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.6, HIGHLY RECOMMENDED UPDATE FOR MAINNET, MANDATORY FOR NEW, RESET TESTNET AND ETHEREUM BRIDGE (GOERLI) USE

v0.9.6 RESETS TESTNET AND COMPLETES ALL PROTOCOL UPDATES FEATURES AND CAPABILITIES EXPECTED FOR THE MAINNET PBAAS UPGRADE. WE DO NOT HAVE ANY NEAR TERM PLAN FOR ANOTHER TESTNET RESET AFTER THE v0.9.6 UPGRADE.

IF YOU HAVE USED TESTNET WITH VERSIONS PRIOR TO v0.9.6, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED IN THE RELEASE NOTES LINKED BELOW.

What’s New

For mainnet, v0.9.6 fixes the following issues:

- An error in fee calculation that can result in nodes having difficulty accepting large, complex transactions with many z-outputs into their mempools.
- Marketplace offers were not showing up properly under getoffers.
- When using the `-privatechange` wallet option with a private change address, attempts to make marketplace offers could fail.

For testnet, v0.9.6 adds the following:

- Caching and performance improvements for advanced PBaaS functions, providing enough performance to keep high transaction bandwidth and/or cross-chain throughput under load for each chain.

- Most hardening required for the mainnet release.

- **The Verus/Ethereum Bridge!** v0.9.6 includes the upgraded Ethereum bridge to the Goerli testnet live with all bridgekeeping, Ethereum Gas price tracking, and reward incentive capabilities for miners and stakers. Rewards for bridgekeeping are delivered both on Verus and on the Ethereum blockchain. Access the Ethereum bridge website (https://ethtest.verus.services) to move over some Goerli Ethereum or USDC, and participate in the Bridge.vETH liquidity basket launch to hold and earn fees on a basket with vETH, USDC, and VRSCTEST used as the fee converter for all cross chain transactions between Ethereum's Goerli and VRSC testnets.

v0.9.6 also includes support for new parameters `-arbitragecurrencies=["VRSCTEST","vETH"]` and `-arbitrageaddress=valid R-address, pubkey, or ID`. They're exciting, but don't try to use them just yet. They are for testing right now, and are preparing for an "auto-arbitrage" feature already supported in the protocol, but which we hope to have in the daemon before mainnet PBaaS activation that combines decentralized on-chain marketplace offers for currency with liquidity basket import processing to provide miners and stakers the opportunity to commit some amount of funds from their node wallet in common currencies to spot and leverage guaranteed arbitrage opportunities when making a block. These opportunities arise between orders on chain and liquidity basket pricing that if leveraged to earn, actually result in P2P crowdsourcing of liquidity for all baskets across the network, both from the blockchain itself as well as miners and stakers, while benefiting all participants in the following ways:
- Miners and stakers can capture guaranteed arbitrage opportunities and arbitrage with any currency that they choose from their wallets.
- Blockchain users can set limit orders and have them filled if they are near the market price for a currency, or if the market moves in their direction.
- LPs and users of liquidity baskets for conversion get as close to market prices as possible on their conversions, never releasing funds from a basket before an on-chain, MEV-resistant auction takes place for the pricing of those assets.

Although this release is likely to be robust enough for a mainnet, we have come this far with quality and will not rush for a few days earlier upgrade. We still plan to review and complete a small number of hardening items before PBaaS 1.0 is ready for its mainnet release.

That said, this release fixes all reported issues in all community testing. Community developers, including myself are grateful for all of the testing many have contributed to to date. It has made it possible to get incredible test coverage across all network capabilities, and we appreciate all of your participation and pushing of boundaries. This release is expected to be the final PBaaS protocol that will be released on mainnet. There are no known changes required, and all functions should work as intended.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Verus PBaaS and DeFi RC6 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don’t expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the

same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128                   // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

  "name": "PBaaSChain",
```

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults


```
  {
```

```
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
  ],
  "initialsupply": 3000000
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/20ea5429375c6c1a82a27d6de31a480187b97a52eec86aea8f4e8f6789df8e89/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/e881b3fc7d5de1a1e69da3722ac6202bf8c4e4ff8722c9cc2f0f5c587a8a0dd7/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/316a831789cb3e0cc878bab5d847c9b27850ce40feee828bc564373f2b1f59b2/detection
Windows:
https://www.virustotal.com/gui/file/a61fc935bf180690ab7cc34a8da54e36728acfd11a130476be66204ae251f5a6/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AQRhIwABQSCdvpJmvU7+i9gzPNBtksrPF4IFbPp+F/AfhjIwABQR+53tZoAd2tV1l/
XV+Gos9acojPtHrtipWpdpQXKWqS+yRRzXA++Fec/OXT/XJpfa73IUojUhXu7C9t0X2/ldIt"
"/Downloads/Verus-CLI-Linux-v0.9.6-x86_64/Verus-CLI-Linux-v0.9.6-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AQRhIwABQSCdvpJmvU7+i9gzPNBtksrPF4IFbPp+F/AfhjIwABQR+53tZoAd2tV1l/
XV+Gos9acojPtHrtipWpdpQXKWqS+yRRzXA++Fec/OXT/XJpfa73IUojUhXu7C9t0X2/ldIt"
2f611ce3d27cadc265f1088508f3dcb2e9262acbcbe8b20bcddc661138303be1
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.5-2, RECOMMENDED UPDATE FOR MAINNET - CRITICAL FOR CONTINUED TESTNET USE

YOU MUST BE UPDATED TO v0.9.5 OR LATER BY BLOCK 2291830, AT APPROXIMATELY 11:00 PM UTC, SATURDAY, NOVEMBER 19th, 2022 TO RELIABLY STAY SYNCHRONIZED TO THE VERUS MAINNET. WHILE THERE MAY BE A MAINNET PBAAS UPGRADE BEFORE THAT DATE, WE RECOMMEND THAT IF YOU HAVE NOT UPGRADED YET, THAT YOU DO SO NOW.

As with all v0.9.5-* releases, this release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022.

What’s New

For mainnet, this release labels spendable transactions for the getaddressutxos RPC API, which is used by Verus Mobile. This update is critical for users that have not upgraded to v0.9.5+ before block 2291830.

For testnet, v0.9.5-2 introduces minor notarization fixes that will be important for those intending to launch and witness PBaaS chains.

Ethereum Bridge

Although its deployment has been delayed, there are no technical blockers, and we still expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services/, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the pbaas-development channel, help the community prepare for the imminent mainnet release of the PBaaS upgrade and start building your vision today!

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don't expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

```
    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {
```

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/efd0d50b9bc15dfec9efa1e0d4d230cf283eff2939f1eaae543306fd26b728ea/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/4f21cafbd0bdf17f38db11467701ca771af846c122eeee6531ad07de2a6371e2/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/e55fa0ed9121c15cbe39d0fe2181d0ee8402540b242f0736b22b4bd497b2eb35/detection
Windows:
https://www.virustotal.com/gui/file/5bcc58941c85243744be4ff7cf053c9fa0a9d3bd64afd572bd1cb5542bcf3438/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AUbeIgABQSCwKpDUADm6FXRioPN0/hUdFHhMTSolY9cgJC6P7w65KR31IKjCDkgr60IWrTRokUpYR8X
97BAvZkcgKZYmPbnA" "/Downloads/Verus-CLI-Linux-v0.9.5-2-x86_64/Verus-CLI-Linux-
v0.9.5-2-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AUbeIgABQSCwKpDUADm6FXRioPN0/hUdFHhMTSolY9cgJC6P7w65KR31IKjCDkgr60IWrTRokUpYR8X
97BAvZkcgKZYmPbnA"
e3ad1a6b967ed8ce177e812409d158f8dd82039e8f994e96eddf1f35f3cc7940
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.5-1, CRITICAL UPDATE FOR MAINNET MANDATORY FOR CONTINUED TESTNET USE

As with the last release, this release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022. YOU MUST BE UPDATED TO v0.9.5 OR LATER BY THEN TO RELIABLY STAY SYNCHRONIZED TO THE VERUS MAINNET. WHILE THERE MAY BE A MAINNET PBAAS UPGRADE BEFORE THAT DATE, WE RECOMMEND THAT IF YOU HAVE NOT UPGRADED YET, THAT YOU DO SO NOW.

Please make sure that if you do choose to wait for the possibility of a mainnet PBaaS upgrade before this activation that you do upgrade to the latest release before Saturday, November 19th.

What’s New

For mainnet, this release has no material changes over v0.9.5, but is still considered critical, as mainnet users should be upgraded to v0.9.5+ before block 2291830.

For testnet, v0.9.5-1 fixes an issue discovered on testnet where duplicate ID exports blocked cross-chain processing temporarily and also introduces a protocol improvement designed to make the Verus anti-MEV DeFi protocol even more resistant than what we believe was already the leading algorithm to multi-block attempts at MEV. The protocol change makes the following adjustments:

Conversions and cross-chain transfers still roll-up into exports that are collected from one or more blocks of transfer transactions, but to determine when a block is the last of the prior export or first in the next requires a random bit pulled from later blocks.
The last miner or staker to complete a block of an export retroactively receives 10% of all fees from processing that export, including all conversions, once the rollup is processed as an import. This reward used to go to the miner or staker who processes the export, which is no longer the case.
Import transactions can now include up to n/2 arbitrage transactions, where n is the number of reserves in a liquidity basket currency.

These changes together serve to disincentivize any miner or staker from attempting to exclude a large set of transactions from a block and front run all of them, as they will have an even chance of having all of those excluded transactions put back in with their transaction, turning potential gain into potential loss. They would be better off saving their working capital to arbitrage imports by joining transactions, bringing them close to market, and earning money without risk of loss. In addition, if they do end up putting more value of transactions in a block that caps the export rollup, they will reliably earn meaningful rewards without risk of loss in that way as well.

Ethereum Bridge

Although its deployment has been delayed, we expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don’t expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the

on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,


    0,


    1000000


  ],


  "initialsupply": 3000000


}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/2d168b0c0e850e7901b80b7d0b05516f03e705e9b8c78f0aa70ac347bc4161e7/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/de0b4d0eb4230858346d68d5437b97cd26204e4a931140ec470add1fdb1edb56/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/1536642a093b525bd90ea8efccffb7a516888d323c1fb3c5a240fd11afb621bd/detection
Windows:
https://www.virustotal.com/gui/file/4d2705506c869bb7c5aade81ac09f29b9b67c9cb790501bdd56933f509fcfdbd/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AdzNIgABQR/6S5TFXXCNHwfznbswtSshmEbbze3i/APbgh4OYeCVECZ/ajii2GUFRGGrxfm5ZHtZ2ME
PhWeyH0QcfUIymLqz" "/Downloads/Verus-CLI-Linux-v0.9.4-3-x86_64/Verus-CLI-Linux-
v0.9.4-3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AdzNIgABQR/6S5TFXXCNHwfznbswtSshmEbbze3i/APbgh4OYeCVECZ/ajii2GUFRGGrxfm5ZHtZ2ME
PhWeyH0QcfUIymLqz"
eeaf6b6e93226d4d1f82ce544cebe780f0376eee93c8300c84a7329c55f72a8a
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.5, CRITICAL UPDATE FOR MAINNET AND TESTNET

This release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022. You must be updated to v0.9.5 or later by that time to reliably stay synchronized to the Verus mainnet. We are also working to get a mandatory update release during these next couple weeks, which will have the Verus PBaaS mainnet upgrade finalized as well. If you would prefer to wait for the PBaaS mainnet upgrade and upgrade once, we cannot guarantee that the PBaaS mainnet release will be available in the next couple weeks, but we are working to make that a reasonable choice as well. Please make sure that if you do choose to wait to upgrade that you do upgrade to the latest release before Saturday, November 19th.

New Signature Capabilities on Mainnet & Testnet

In addition to continued support for all of the `signmessage`, `signdata`, and `signfile` APIs, as well as their verify* counterparts, v0.9.5 introduces 2 new APIs and CLI commands, `signdata` and `verifysignature`, that encompass all of the capabilities of previous signing and verification APIs and add new capabilities for advanced signatures and verification. These new advanced capabilities were designed for community work by @allbits, designing a more powerful and flexible framework for NFTs and ID-bound legal rights and agreements. In addition to digital signing of documents or information using an ID, the new APIs enable easily binding together multiple sources of content, documents, identities, and VDXF keys together into one verifiable signature and then being able to verify that signature, including all of its specific bindings.

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don't expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Emoji IDs and 🎃Chain

In the spirit of this release day, we have taken advantage of VerusIDs and currency emoji support and created a new, merge mineable PBaaS chain, 🎃. It has been said that those who merge mine 🎃chain at midnight will disappear into the crypto-metaverse, never to be seen again, become VerusID avatars, and be forever doomed to being bought and sold on the VRSCTEST marketplace for all eternity (or until the next testnet reset). YOU HAVE BEEN WARNED!

We expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,

    0,

    1000000

  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/282f6fc1bd95ddab588dd7fbc088372404eb0f7a746633f0493a2c446fb037b9/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/6bec22fc8fa9c19a6d1b7771a7f85d2754ddf8d2c67202c3c66052d0a08f7311/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/2e1798c5e942feec9634b418db6bec94385bb47ba6892ea3af6bb88122103b92/detection
Windows:
https://www.virustotal.com/gui/file/20252ebb7224d51ba6048b44524b305b3522bb34f38b52424f507148a2fb463b/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"ASKRIgABQR8y/Iv9s1rJfeLS8Kdq4SxnKKFtTFXjJ/Di7QKwiejX/0WRyEl9UWUa0W4+zKiyW7fMGcG
ZjPm62lrpPLs8sG/P" "/Downloads/Verus-CLI-Linux-v0.9.4-3-x86_64/Verus-CLI-Linux-
v0.9.4-3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"ASKRIgABQR8y/Iv9s1rJfeLS8Kdq4SxnKKFtTFXjJ/Di7QKwiejX/0WRyEl9UWUa0W4+zKiyW7fMGcG
ZjPm62lrpPLs8sG/P"
092e4fbc470530a7943fe080c27d9470483ea7947400d6da599a2ae3449b7ec9
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4-5, RECOMMENDED FOR MAINNET AND TESTNET

This release fixes an RPC API issue on mainnet and testnet that can cause a daemon fault when the getaddressmempool API is used. This fault while running is only caused by non-typical API, insight explorer, or CLI use, but if it does happen, may result in the need to rescan or resync your wallet. We know of no one on mainnet affected by this issue. We recommend that all users upgrade.

This release also includes a new capability for testnet, which was pioneered by CHIPS developer @Biz, who worked to modify and test the Verus PoP algorithm to support 10 second block times, which the CHIPS chain uses to enable its game application focus. These new parameters will enable the CHIPS blockchain to move to being an interoperable, merge mineable PBaaS chain with a 10 second blocktime when PBaaS goes live on mainnet. To learn how to test the new CHIPS test chain on testnet, visit the #pbaas-development channel, as the support in v0.9.4-5 is not as complete as it will be in the next and hopefully final testnet reset. In that next version, blocktime and the pow averaging window for any PBaaS chain’s PoP consensus will be able to be configured between 10

seconds and 2 minutes in addition to its notarization frequency as part of the chain definition, enabling a greater variety of PBaaS use cases, transaction throughput, and application capabilities.

As a reminder, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services , using your metamask, converting or sending cross chain through the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an

unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the

same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128                   // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

  "name": "PBaaSChain",
```

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults


```
  {
```

```
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
  ],
  "initialsupply": 3000000
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and

associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/037f6aa4d508c8a5d34cf064d27f69b56e3d3825405087a3244914cba40cb254/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/e271171db246c15a3bc17f66299948eb81f451fcc84308ae0c1e518d57ee7aa5/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/103cea57b996273b85efdeb79625df84e2a8de8a3929a78b89b75d0f072677e4/detection
Windows:
https://www.virustotal.com/gui/file/18cf5a62e066b90b683558194d9f3dbc2a95d12cddabf0d9e29e128c06139ac1/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AZHCIQABQSBJ+t3wHNYyoWu42QrK3ncJKPdppvC2shhwDlPPR/ZKCwvu+Y8LEy8F/
Wyj23ba9grVsJZQu4BCHDicQny6vIjt"
"/Downloads/Verus-CLI-Linux-v0.9.4-3-x86_64/Verus-CLI-Linux-v0.9.4-3-
x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AZHCIQABQSBJ+t3wHNYyoWu42QrK3ncJKPdppvC2shhwDlPPR/ZKCwvu+Y8LEy8F/
Wyj23ba9grVsJZQu4BCHDicQny6vIjt"
413bf64c365c3fcf92c273d1a8674626284e340437fc0634872b9d0ee483f327
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4-4, UNCHANGED FOR MAINNET -- RECOMMENDED FOR CONTINUED TESTNET USE, MANDATORY FOR CONTINUED TESTNET MERGE OR BRIDGE MINING

This release has no material mainnet changes.

This release primarily upgrades the ETH bridge and also fixes a PBaaS cross-chain notarization issue that made miners and stakers consider themselves ineligible to enter cross-chain notarizations. There are no actual protocol changes.

The ETH bridge upgrade introduces the ability to enter notarizations between Ethereum Goerli and VRSCTEST using bridgekeeper without an Ethereum private key in your conf file. This will enable you to still mine and potentially earn by agreeing or disagreeing with notarizations mined into the Verus blockchain, without needing to make transactions on or spend Ethereum for fees. This should also work with a free Infura account.

As a reminder, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services , using your metamask, converting or sending cross chain through the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. Drop by the #pbaas-development channel to help us prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

```
    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {
```

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/81692fb40d8bad1b92615884494be36a6286f0321f49f91a5f19107f61d60314/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/048fd619ed06a2002186a09f85f9dc56c6f0d3033c164b12384fd051537d05c0/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/3c178229023bcfa54579a9d2ee661696f280f46a9cab1a99f9ff0dac7d366be1/detection
Windows:
https://www.virustotal.com/gui/file/094560c238f21e778593fd756406773adec38ae5bc6a3770c0f9662c62e6cb27/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AfmaIQABQR8yN3LyVymt/ZRYvyuGTma3rEbxFshyLOa3zI5qAXTaz34gxj30QXiluQ5TJJ7jxQwJaLU
l6bJLMvfNvlzI+KBw" "/Downloads/Verus-CLI-Linux-v0.9.4-3-x86_64/Verus-CLI-Linux-
v0.9.4-3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AfmaIQABQR8yN3LyVymt/ZRYvyuGTma3rEbxFshyLOa3zI5qAXTaz34gxj30QXiluQ5TJJ7jxQwJaLU
l6bJLMvfNvlzI+KBw"
7cea478ac61c8328fea9a42f8744f9112cf6723aa0f39c727c1d090f5daa5b34
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4-3, UNCHANGED FOR MAINNET -- MANDATORY FOR CONTINUED TESTNET USE

This release has no material mainnet changes.

This release upgrades the testnet protocol to fix an issue with the Ethereum bridge and VRSCTEST mining and staking. Please update to v0.9.4-3 to continue mining and staking on testnet and the Ethereum bridge. This version also fixes an issue discovered when defining Ethereum NFT mapped tokens, which are not yet supported in the released Ethereum contracts, but are updated no in the daemon and ready to use when the contracts are upgraded, hopefully in the next few days.

NFT mapped tokens will allow you to define an Ethereum NFT as the token that controls revoke and recover for an ID. Similar to how you can define a token for tokenized ID control, an NFT mapped token, when defined, refers by contract and tokenid to a specific NFT on the Ethereum blockchain. Once defined as the currency for a specific ID or sub-ID and linked to the Ethereum NFT, that NFT will be able to be sent over from the Ethereum blockchain as a satoshi of that currency, sent to any address or ID on the Verus blockchain, exchanged trustlessly on the Verus on-chain P2P marketplace, and used to revoke and recover the ID which defined the token. Once the contracts are updated, you will be able to mint NFTs on Ethereum's Goerli testnet, and endow your Ethereum NFTs with all of the power of your specially named VerusID. This new technology allows you to protect your NFTs with VerusID's revoke and recover technology, bind your NFT to the pseudonymous ID you use to log into applications, use it to extend and publish much more than just the NFT, and generally take your Ethereum NFTs to the next level!

The rest of these release notes are the same as the prior release. Drop by the #pbaas-development channel to help us prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,

    0,

    1000000

  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/623213f8bd49817d1e5196f41852846164caa93e2baad3c715878d390d90fe22/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/286ebb51f7187b2c023c140e33eb3e0d95c9ddf6ad42256c603c5cc400cd8a0f/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/245ef69a3cf4a0035eff1bed792817fbd1b7dd0a23a4c6a3506637fef20b40b4/detection
Windows:
https://www.virustotal.com/gui/file/2345a09fd73a0f4218584af58ac164ffdd2b8a9a14bda15af49f25a4188c649a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AfqOIQABQR9JBWMw2o1wOS6qAHZW8D9cm0MfDSY2EZyleWFw3JbXJBnbcw0Pa4FFLeNw+kJVZPQfvsA
2fEW2hukvCYly1GDb" "/Downloads/Verus-CLI-Linux-v0.9.4-3-x86_64/Verus-CLI-Linux-
v0.9.4-3-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AfqOIQABQR9JBWMw2o1wOS6qAHZW8D9cm0MfDSY2EZyleWFw3JbXJBnbcw0Pa4FFLeNw+kJVZPQfvsA
2fEW2hukvCYly1GDb"
f5d6e69708f341dac3b8170dd25083690ce6585f84c4785cbcf975560ddc21a4
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4-2, UNCHANGED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release has no material mainnet changes.

THE NEW ETHEREUM BRIDGE IS EXPECTED TO BE DEFINED WITHIN THE NEXT 24 HOURS AND FOR THE BRIDGE LIQUIDITY BASKET TO LAUNCH SHORTLY AFTER DEFINITION. IF YOU LEAVE A PRIOR VERSION NODE RUNNING UNTIL AFTER THE VETH BRIDGE IS DEFINED, YOU MAY FORK AWAY DUE TO CHANGES IN THIS RELEASE AND THEN NEED TO RESYNC TO TESTNET FROM SCRATCH TO CONTINUE.

For testnet, this is RC4 of the Verus PBaaS protocol, which includes some Ethereum bridge related fixes and a merge mining API change. With the API change, all merge mining nodes must upgrade to the new protocol to be compatible with each other and participate in cross-chain notarization rewards.

After the Ethereum bridge has been defined, you will be able to send currencies over to Verus testnet from Ethereum's Goerli network and also participate in the launch of the Ethereum bridge liquidity basket with VETH, USDC, or VRSCTEST by using `sendcurrency` with `"preconvert":true` on the command line or through the GUI. We recommend that you get some Goerli ETH using this facet or others (https://goerli-faucet.mudit.blog/), which will make your experience of all Verus PBaaS can offer a bit more well rounded when the Ethereum bridge is available once again.

Once the Goerli Ethereum liquidity basket (Bridge.vETH) activates on testnet, which is a 100% backed basket composed of 1/3rd each of Verus, Ethereum, and USDC (all testnet tokens), you can use it from Metamask on Goerli, from the Verus CLI or GUI on the vrsctest chain, or from CLI/GUI on other PBaaS chains to provide liquidity while holding Bridge.vETH, a 100% backed basket of three currencies of balanced relative value, or to pass through and convert currencies between Verus, Ethereum, USDC, and Bridge.vETH.

The Brige.vETH currency also coordinates with the vETH@ gateway and enables registering `*.veth@` IDs for $5 worth of Bridge.vETH, based on its internal USDC conversion price at the time of registration. When you register `*.veth@` IDs, 0.02 VRSCTEST goes to miners and stakers, and $5 worth of Bridge.vETH is burned, reducing the total supply of Bridge.vETH for everyone without reducing reserves, and contributing to the relative value, "real yield" of Bridge.vETH, relative to its three reserve currencies. This is the way that all decentralized, fractional currencies which enable ID registration accept registration fees, equally benefiting all fractional currency holders/liquidity providers when IDs are registered, while providing Verus miners and stakers the identity import fee of 0.02 VRSC.

`*.veth@` IDs have the ability, along with root vrsctest IDs, to make 1:1 ERC20 mapped currencies on Ethereum. They can also be used to create Tokenized ID control tokens (see notes for v0.9.4-1), as all types of IDs can, but if they are used to do that, they will have used their currency creation ability for that rather than an ERC20 mapped currency.

The rest of these release notes are the same as the prior release. Drop by #pbaas-development to start building your vision on Verus and help prepare PBaaS for mainnet!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set
```

```
OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals



OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency



OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128                   // if set, this routes external
currencies



OPTION_PBAAS = 256                     // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512         // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,

that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1
```

```
],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

"networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
  {

  "currencies": [

    "VRSCTEST",

    "PBaaSChain",

    "USD"

  ],

  "initialcontributions": [
```

```
    380228.12033701,

    0,

    1000000

  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/3ed2631d4cffb267afd1e5497b3eca3e23bdb8289dac0cdec9da12dbe03af969/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/6b852efe7406cbd9e9ed530d70721fec96cad53bf169cf96c6aa173d346c2349/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/8a573fe7db3bb29ff67de938e8d7d474daa1a75d5ce81a2d385233fefc0b8e8f/detection
Windows:
https://www.virustotal.com/gui/file/0032e7e1cd6239f4521f0f48b56339511a32555475224c266f4babe161278c24/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AcBSIQABQSBIKAh7/OaCTKSTt5djpmiQ7Mx8owKFNPdolko/8I4WB3StZhvchKRFfAAo5bV9eREGNr5
Vq4z+4Fwl+ZRkw1Al" "/Downloads/Verus-CLI-Linux-v0.9.4-2-x86_64/Verus-CLI-Linux-
v0.9.4-2-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AcBSIQABQSBIKAh7/OaCTKSTt5djpmiQ7Mx8owKFNPdolko/8I4WB3StZhvchKRFfAAo5bV9eREGNr5
Vq4z+4Fwl+ZRkw1Al"
ce611f081daba33876af496c5699b669da0e35343106801e21c4ed6169460140
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4-1 with the world’s first self-sovereign, tokenized ID control technology, RECOMMENDED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release has no material mainnet changes.

For testnet, this is RC3 of the PBaaS protocol and is required to follow the main testnet chain. RC3 has some significant and important API additions, including the ability for any ID at all, even sub-IDs, to generate a single token as its currency. That token then enables whoever holds it to have another level of revoke/recover power over the ID, in addition to the normal revocation and recovery authorities, which also still work. This technology is now available in the pre-existing APIs, updateidentity (for changing revocation / recovery authorities), revokeidentity, and recoveryidentity. Please see help for details.

In addition, we have continued to harden the protocol and fixed issues with cross-chain currency definition import/export and the new ability to burn reserves into a liquidity basket that were reported in the current testnet.

The rest of these release notes are the same as the prior release. Drop by pbaas-development to start building your vision on Verus while helping get PBaaS ready for mainnet!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the

same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128                   // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024   // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

  "name": "PBaaSChain",
```

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults


```
  {
```

```
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
  ],
  "initialsupply": 3000000
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and

associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/a202cdbdde620f9106a6ea31421f5f948f13f18ad1fa872bd64bc1e18c7b9ddd/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/eca39c1e72e71fb50dae2d6e30f1d24ca571913d4dc4c238bcf087d3983ceb6b/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/3c03bbfd007fd2bf278877bc37837232113d522479a752157d522790b0afc093/detection
Windows:
https://www.virustotal.com/gui/file/43915c286e9d37e220444dbd2f3d8e56ee3cadfa64f61885670fb7bc8a0a5fe4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"Ac1PIQABQSCaobx7IQFxa4N001E/ZVHFLAlhTrJ9iJeDB+l51VvFuwXlaH+mEtfavIJLqgnALH20Ul3
zkV2/Iq2EvZhjsMno" "/Downloads/Verus-CLI-Linux-v0.9.4-1-x86_64/Verus-CLI-Linux-
v0.9.4-1-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"Ac1PIQABQSCaobx7IQFxa4N001E/ZVHFLAlhTrJ9iJeDB+l51VvFuwXlaH+mEtfavIJLqgnALH20Ul3
zkV2/Iq2EvZhjsMno"
b70d6c8a0cc5d13cdcb03ad0599b960b80a596c56fdaf548c7cac04f9f89cbb9
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.4, HIGHLY RECOMMENDED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release fixes a mainnet issue that may, in rare cases, cause a node to deadlock.
It also adds the following new features to mainnet:
- Auto TLS encrypted connections between nodes on mainnet and testnet, no action needed, but you can require that all connections be encrypted by starting the daemon with “-tlsenforcement”.
- A new “-idindex” flag that enables 3 new APIs to lookup identities by their primary addresses or revocation and recovery addresses, `getidentitywithaddress`, `getidentitywithrevocation`, and `getidentitywithrecovery`.
Generally improved multithreading within the daemon.
- These improvements allow anyone to now write applications to easily graph or display public relationships between identities, transparent addresses, and transactions without compromising the privacy of private addresses, messages, or transactions between identities using name@:private endpoints, z-addresses or communicating via off-chain channels.

For testnet, this is RC2 of the PBaaS protocol and has some significant and important advances and changes. In addition to a great deal of progress hardening for mainnet, v0.9.4 introduces the following new features to the PBaaS testnet:
- All features above for mainnet
- NFT mapped tokens and tokenized ID control - In v0.9.4, as with v0.9.3, you can use root identities to create any kind of currencies, *.veth IDs to create Ethereum mapped currencies, and what is new in v0.9.4 is that you can now use any type of ID or sub-ID to create a new kind of currency that will always have a 1 Satoshi supply and enable anyone holding it to have power over the ID that issued it. This type of currency is called tokenized ID control and can be treated like a super NFT over all copies of the source ID on all chains that were exported to other chains after the currency was created. Once the Ethereum contracts are upgraded with this capability,* .veth or IDs with root names on the Verus blockchain can be used to create Ethereum NFT mapped currencies, which enable a Cryptopunk, Ethrock, or whatever other NFT you’d like to not only be sent to Verus, held on a revocable/recoverable ID, and also permanently upleveled with ID powers, being cryptographically bound as an ID’s permanent control token that can be sent to any chain or to Ethereum and back.
This unique Verus-only technology and VerusIDs enable new models of complex, atomic and fully decentralized transactions. By using exported IDs across any number of networks and an ID control token, ownership of an unlimited array of assets across an unlimited number of networks can for the first time ever, have ownership transferred in one transaction.
- While NFT mapped tokens are supported on the RC2 PBaaS protocol, they have not yet been implemented in the Ethereum contracts. You will not be able to send NFTs from Ethereum to Verus

or PBaaS chains until that work is complete, which we do not expect to take a great deal of time.
- Liquidity currencies now support new methods to burn currency with the following results:
- `burnweight` - available to centralized currencies only and enables burning the primary currency to raise the reserve ratio weights of all reserves.
- `burn` for reserve currencies - in prior releases, burn was only used for primary currencies and would reduce the supply of a currency relative to its reserves, increasing the on-chain value of the primary currency. In v0.9.4, you can now also “burn” reserve currencies in a liquidity basket currency, which simply donates that amount of reserve into the currency’s reserves without taking any new supply in exchange. This is available to decentralized or centralized currencies.

- Bridgekeeper is now available for permissionless bridge notarization and selectable in the GUI on the mining page. For CLI, we will make a Node application available for testing in the next few days.

- RC1 cross-chain witness and notarization protocol enhanced in RC2 as follows:

    - In RC2, any notarization to be confirmed must be first posted publicly to allow for potential to prove invalidity, and then later confirmed. In the prior protocol, a witnessed notarization could be posted to a chain and accepted when first seen. Upon analysis and discussion with @allbits and others, we realized that the attack vector of even stolen notary/witness keys combined with an in-secret 51% combined hash/stake attack could be prevented if we changed the protocol to first require posting the ready to finalize data and allow notaries who recognize that there is a signature on chain for something they don’t agree with (likely stolen key), to simply revoke their identity, which will prevent that notarization from ever being confirmed. This same security feature is supported between all PBaaS chains and has been implemented in the Ethereum contracts for witnesses as well. This new protocol also provides for other types of evidence to prevent finalization, such as proof that there exists a more powerful (most power rule) tip than one witnessed. Enabling decentralized, cross-chain most power rules are an advance in cross-chain protocols that over time will strongly protect censorship resistance qualities of any multichain protocol.

    - The above technology required changes to generalize evidence types between chains, enabling cryptographic enforcement that all notarizations are ultimately determined by miners and stakers and finalized by witnesses who must follow most power

(stake+work) rules based on the protocol. Witnesses cannot finalize a chain tip unless alternating miners and stakers have posted 3 consecutive, agreeing, progressing cross-chain notarizations, without intervening disagreement, trading off more certain cross-chain transaction latency for on that depends on miners and stakers, meaning maximum decentralization and a high cost to DoS attack cross-chain, approaching impossible over any significant length of time. This approach makes the incentivized merge mining and bridge mining opportunities, as well as miner/staker participation, automatically part of and also critical to the cross-chain protocol. In v0.9.4, all users on the network may mine the Ethereum bridge via the bridgekeeper tool on the mining page in the GUI.

- With this new protocol, all PBaaS chains on testnet will consider witnessed / finalized tips final and will not unwind past those witness points or follow a chain that disagrees with a known finalized tip. When this protocol goes live on mainnet, and for any PBaaS chain whose community opts to use the Komodo notary service, Komodo notarizations will continue to ensure a stable chain tip, but no matter what other notarizations are present, PBaaS chains will always follow their proof of power protocol and lock to their witnessed tips as well.

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

The rest of these release notes are the same as the prior release. Drop by pbaas-development to start building your vision on Verus while helping get PBaaS ready for mainnet!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1                  // allows reserve conversion using
base calculations when set

OPTION_ID_ISSUANCE = 2                 // clear is permissionless, if set,
IDs may only be created by controlling ID

OPTION_ID_STAKING = 4                  // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8                // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16        // if set, this chain requires
referrals

OPTION_TOKEN = 32                      // if set, this is a token, not a
native currency

OPTION_SINGLECURRENCY = 64             // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128                    // if set, this routes external
currencies



OPTION_PBAAS = 256                      // this is a PBaaS chain definition



OPTION_GATEWAY_CONVERTER = 512          // this means that for a specific
PBaaS gateway, this is the default converter and will publish prices



OPTION_GATEWAY_NAMECONTROLLER = 1024    // when not set on a gateway, top
level ID and currency registration happen on launch chain



OPTION_NFT_TOKEN = 2048                 // single satoshi NFT token,
tokenizes control over the root ID
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to

initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{
```

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

```
    "halving": 0,

    "eraend": 0

  }

],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {
```

```
      "networkaddress": "111.111.111.111:10000",

      "nodeidentity": "Node1@"

    },

    {

      "networkaddress": "111.111.111.112:10000",

      "nodeidentity": "Node2@"

    }

  ],

  "gatewayconvertername": "Bridge",

  "gatewayconverterissuance": 1000000

}
```


The bridge definition has overridable defaults

{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

```
}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/a0bf7673ca1e740f7261fd12888edb8ef68586c48527921f6a5e272cfd6bb57a/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/269a96dd66a50fa670695bf2489eeb7dc837878e7b56a528f1c2de977524c96d/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/de243d36d6dcdb06ef70be1da3b1d58d875192de73b0a37afc6dcbbb614f2908/detection
Windows:
https://www.virustotal.com/gui/file/07f747e596586b9666f2ae74f0865890b4db3fb85e8d94e9e376fbd2393bc37c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
"AYUoIQABQR8xuMk0N7mT6KmiXJ5VKJnZR27+udobzfV0DTlb5S98KC5mYJRITYBta+vRvYtKF2EUCaP
dQZsa+pMSawjEflRN" "/Downloads/Verus-CLI-Linux-v0.9.4-x86_64/Verus-CLI-Linux-
v0.9.4-x86_64.tar.gz"
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
"AYUoIQABQR8xuMk0N7mT6KmiXJ5VKJnZR27+udobzfV0DTlb5S98KC5mYJRITYBta+vRvYtKF2EUCaP
dQZsa+pMSawjEflRN"
8b2c32a13f1154a97644207076c6f73ef774c7897029b1f0d5018677645955b7
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.3, NON-MANDATORY FOR MAINNET -- MANDATORY FOR TESTNET USE

This release fixes one issue on mainnet where a node using -connect=”networkaddress” may still attempt to connect to other nodes.

This update includes the first protocol release candidate for mainnet PBaaS, VerusID protocol upgrades, and Verus DeFi. In addition to some functional fixes and full web site support for mapped currencies and simple commands for mapping Verus currencies to any ERC20 currencies on Ethereum with provable, auditable 1:1 mapping and cross chain send/receive. In addition this new protocol upgrade changes some parameters of the Verus Proof of Power consensus algorithm, enabling smoother starts, restarts, and general operation of PBaaS chains, regardless of their native currency supply or emission schedule. As these consensus parameter changes would require either a testnet fork, which would not be relevant for the mainnet upgrade or a reset, accessing the v0.9.3 testnet requires that you reset your local testnet by deleting the vrsctest and pbaas data folders from the locations described below.

As part of this release, we will be launching the new Ethereum bridge and enable full use of Ethereum mapped currencies on the Goerli Ethereum testnet (note that we were previously using Rinkeby). This makes it easy for any currency definition to become a fully decentralized, auditable stable coin, simply by mapping to DAI, USDT, USDC, or another stable coin on Ethereum. In addition, the new website for Metamask support and use from Ethereum now displays all mapped currency options, even those added when someone exports a currency definition to Ethereum, either as a mapped currency or for using any Verus or PBaaS currency as an ERC20.

This release also resolves an issue with setting max preconversion amounts on currency launches and addresses a number of minor protocol issues found as a result of the active community testing support we are seeing on testnet.

v0.9.3 RESETS TESTNET AND ADDRESSES ALL KNOWN ISSUES FOUND DURING TESTING ON THE NETWORK. FROM HERE, WE HOPE ALL COMMUNITY MEMBERS WILL JOIN IN TRYING OUT ALL FEATURES OF TESTNET TO HELP ENSURE THAT THE FINAL MAINNET RELEASE, WHICH WE ARE WORKING TOWARDS AS SOON AS IT IS CONSIDERED READY AND HARDENED, WILL BE BOTH SMOOTH AND WORLD-CHANGING.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Platform Support

- Debian 9 (Stretch) is no longer supported

PROTOCOL RELEASE CANDIDATE RC1

With v0.9.3, The Verus PBaaS protocol, including networking, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi, is considered release candidate RC1 for the coming mainnet upgrade. The core developer focus now remains on protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Now would be a good time for development capable projects hoping to get a head start on using Verus's transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.3, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {"proofprotocol":2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.

More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at 100 USD worth of ETH to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.3 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals



OPTION_TOKEN = 32                // if set, this is a token, not a native
currency



OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

```
}
```

The bridge definition has overridable defaults

```
  {
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is

considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/e3920777dc40da783d06df5cf58e3a2bec73257a6aa52e32a357697f8e2840fc/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/9ae67dfc464025af9afedaf6b11857b55f459953a376b8bd969a5e2ebb7f61e2/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/3cf73a75019c0b87b8c7551f423e687b38a8ded3d2d8b9a2e6fd6

5184f207417/detection
Windows:
https://www.virustotal.com/gui/file/ebbfbafe759afe9d40a39473bbaedd15ecd12c53d46c1b1f566b104c2af8fe05/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AYGcHwABQSCPnizC1yvgpTQCGjoVGNa6DTnFk7Y5cJk4B+UnS8ctLwtxP3HsTvqwR/
Hf0TAWFqo3pDc9pBHwfnvVj9ddEDhD /Downloads/Verus-CLI-Linux-v0.9.3-x86_64/Verus-
CLI-Linux-v0.9.3-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AYGcHwABQSCPnizC1yvgpTQCGjoVGNa6DTnFk7Y5cJk4B+UnS8ctLwtxP3HsTvqwR/
Hf0TAWFqo3pDc9pBHwfnvVj9ddEDhD
d167fe2800bfe3da70009ff3861cf2eb7415fa6fb7c3e132bde10161ac3f49dd
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.2-3, NON-MANDATORY, HIGHLY RECOMMENDED FOR MAINNET -- CRITICAL FOR TESTNET USE

This update enables full use of Ethereum mapped currencies (Rinkeby when the new bridge launches), which provides the ability for any currency to become a fully decentralized, auditable stable coin, simply by mapping to DAI, USDT, USDC, or another on Ethereum. In addition, the new website for Metamask support and use from Ethereum now displays all mapped currency options, even those added when someone exports a currency definition to Ethereum, either as a mapped currency or for using any Verus or PBaaS currency as an ERC20.

This release also resolves two PBaaS chain launch issues, one that prevented no-reward chains with initial emission from being accepted and one that could allow a chain launch to go forward, even when a co-launched liquidity basket failed to get enough participation. Thanks to increased participation on testnet use, we are getting continued excellent coverage and resolving issues, which are increasingly edge conditions as we go. discovered on both mainnet and testnet and on testnet fixes an error creating transactions for sub identity registrations on fractional reserve currencies. If you already went through the testnet reset process on any version since v0.9.2, you do not need to reset testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID

ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus's transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {"proofprotocol":2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals



OPTION_TOKEN = 32                // if set, this is a token, not a native
currency



OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 264,

"currencies": [

  "VRSCTEST"

],

"conversions": [

  1

],

"eras": [

  {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

```
}
```

The bridge definition has overridable defaults

```
  {
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
  ],

  "initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is

considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/41a5644d70f3cffe92b55f4a810ce5b05432dcdb2bb0d198d1cdb327773139cc/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/f28ba4bbfec1205c9109c8b2bc02d56e5ec221bf0fd0d6061d4dbb1b5b5fc884/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/fb8ef0d3631e524a41d7d26bd9e2cfbf0eda2d27c63cfe3a12ce6ea

18f08941c/detection
Windows:
https://www.virustotal.com/gui/file/1583296955647a1b6d67d396526d975f662cb56c354b878dea5898c8f04979da/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AXQSHwABQR+gAuS5YkkVHiKCAjLwqpTNxAxo3wBjRDcYfQ0SqWenTgIB+HA3UUbJkC7qH5YZamiRrr3w
OmFwdl6z7OPbXBDk /Downloads/Verus-CLI-Linux-v0.9.2-3-x86_64/Verus-CLI-Linux-
v0.9.2-3-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AXQSHwABQR+gAuS5YkkVHiKCAjLwqpTNxAxo3wBjRDcYfQ0SqWenTgIB+HA3UUbJkC7qH5YZamiRrr3w
OmFwdl6z7OPbXBDk
09acebdf580f34b322561c247ebbb60a1c975453d4766abbec9229558201a96e
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.2-2, NON-MANDATORY, HIGHLY RECOMMENDED FOR MAINNET -- CRITICAL FOR TESTNET USE

This update improves sync speed in some edge cases on both mainnet and testnet and on testnet fixes an error creating transactions for sub identity registrations on fractional reserve currencies. If you already went through the testnet reset process on any version since v0.9.2, you do not need to reset testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
    - Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
    - Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set

OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID

OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals

OPTION_TOKEN = 32                // if set, this is a token, not a native
currency

OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

    "name": "PBaaSChain",

    "options": 264,

    "currencies": [
```

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

```
],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },
```

```
      {

        "networkaddress": "111.111.111.112:10000",

        "nodeidentity": "Node2@"

      }

    ],

    "gatewayconvertername": "Bridge",

    "gatewayconverterissuance": 1000000

  }
```


The bridge definition has overridable defaults


```
  {

    "currencies": [

      "VRSCTEST",
```

```
      "PBaaSChain",

      "USD"

    ],

    "initialcontributions": [

      380228.12033701,

      0,

      1000000

    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
```

```
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternamе":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/8c8c5240270807e16e8975032f3de5853daaddc76ff667af279ef7df05418082/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/28b726f52f306e6c86e04563dbe7d385d3021d8acb2e0be1d586416b9d83a0f5/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/5908d9acfd3b6d2f65071dd650732f7f38f96ab9e822d90676a3f7f561821c57/detection
Windows:
https://www.virustotal.com/gui/file/0168e46dc9edeb65af91fe5f8842516800c60f2ea793f7cef695232c8bdcaba7/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQn/HgABQR8IlODV3COH5wvsCidoZiBgBc+riVxS4jgdH4WlfJ5TnU03J9UhgJNhNG3VmqpyfNO53JIF
GuYXHjEAgagdeYnO /Downloads/Verus-CLI-Linux-v0.9.2-2-x86_64/Verus-CLI-Linux-
v0.9.2-2-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AQn/HgABQR8IlODV3COH5wvsCidoZiBgBc+riVxS4jgdH4WlfJ5TnU03J9UhgJNhNG3VmqpyfNO53JIF
GuYXHjEAgagdeYnO
4758a04c8d9923cb7cd784cb2f91dcbcc2c38a5f95312406c097ff5b8ba30dc9
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.2-1, NO FUNCTIONAL CHANGES TO MAINNET FROM v0.9.2 -- CRITICAL FOR TESTNET USE

This update fixes an error in creating transactions for sub identity registrations on fractional reserve currencies and it fixes a synchronization issue that could prevent some clients from fully syncing testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it

will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the

same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set

OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID

OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking

OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals

OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals

OPTION_TOKEN = 32                // if set, this is a token, not a native
currency

OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency

OPTION_GATEWAY = 128              // if set, this routes external
currencies
```

```
OPTION_PBAAS = 256                // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

    "name": "PBaaSChain",

    "options": 264,

    "currencies": [

      "VRSCTEST"
```

```
],

"conversions": [

  1

],

"eras": [

  {

    "reward": 1200000000,

    "decay": 0,

    "halving": 0,

    "eraend": 0

  }

],
```

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

```
        "networkaddress": "111.111.111.112:10000",

        "nodeidentity": "Node2@"

      }

    ],

    "gatewayconvertername": "Bridge",

    "gatewayconverterissuance": 1000000

  }
```


The bridge definition has overridable defaults


```
  {

    "currencies": [

      "VRSCTEST",

      "PBaaSChain",
```

```
      "USD"

    ],

    "initialcontributions": [

      380228.12033701,

      0,

      1000000

    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternamе":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
```

```
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/b236e84af64c14fe576ad82370faec59dc3e2ba1df6a587fc3317560491be602/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/13594d432962e107130ab4dccda0241456b1d94b4116232825fe2bf254415981/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/ab1f35048024838bc61627051a6bcd908df02b16327f4c177a31c47d536e1f2b/detection
Windows:
https://www.virustotal.com/gui/file/bcd62e56dce40060a333e81bd2ce2ee87bfb9e41fa1494e0779924e29b5800f8/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AYi7HgABQR/3O5kb1Dyxwwt1tNJL6s7LU9W/griFa/zzeUechZJSaRFsVSYN9btmznD3zfEAoCnDDQiD
2OoB2MwmrdhgDSJI /Downloads/Verus-CLI-Linux-v0.9.2-1-x86_64/Verus-CLI-Linux-
v0.9.2-1-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AYi7HgABQR/3O5kb1Dyxwwt1tNJL6s7LU9W/griFa/zzeUechZJSaRFsVSYN9btmznD3zfEAoCnDDQiD
2OoB2MwmrdhgDSJI
60c686897e4d29a2d73f31f84e01670c6ca912f0b1e6ed9124eae479ad8bd292
```

The result will be true if the signature is valid.

```
true
```

Announcing v0.9.2, OPTIONAL, HIGHLY RECOMMENDED FOR MAINNET, MANDATORY FOR TESTNET USE - THIS IS A BIG ANNOUNCE, BUT IF YOU CARE ABOUT THE PBAAS UPGRADE, WE RECOMMEND READING IT ALL.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID

ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus's transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {"proofprotocol":2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
    - Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
    - Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

`

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
```

```
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals



OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals



OPTION_TOKEN = 32                // if set, this is a token, not a native
currency



OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency



OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

  "name": "PBaaSChain",

  "options": 264,

  "currencies": [

    "VRSCTEST"

  ],

  "conversions": [

    1

  ],

  "eras": [

    {
```

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

```
"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },

  {

    "networkaddress": "111.111.111.112:10000",

    "nodeidentity": "Node2@"

  }

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000
```

```
}
```

The bridge definition has overridable defaults


```
{
  "currencies": [
    "VRSCTEST",
    "PBaaSChain",
    "USD"
  ],
  "initialcontributions": [
    380228.12033701,
    0,
    1000000
```

```
    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is

considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/a70cd7e14f4b43a0c0123fb938881ccc7900396ce14057e4887986c591a2d506/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/26af0cebea1b579b1e7d55c3fcd927a2da1d58c4b3d4461e7039858f3f12f0f5/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/ef820935c721c11b0843e7925be5ba56b77937030028ca4ccab03

8c3117571a1/detection
Windows:
https://www.virustotal.com/gui/file/9c9d63e43e93b6c2b86a1ef2c70d62c92c9b280fa86a1c7c091b4053f48300a5/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AXKyHgABQSAZDjfHQ0HpRG4ORdPljczZMSDrmKVHLHy9YCAiy2wQOgW2wfl8+qJ3BbL3A4pyGyS0WDUT
I7CN7LZNGxjHd47Z /Downloads/Verus-CLI-Linux-v0.9.2-x86_64/Verus-CLI-Linux-
v0.9.2-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AXKyHgABQSAZDjfHQ0HpRG4ORdPljczZMSDrmKVHLHy9YCAiy2wQOgW2wfl8+qJ3BbL3A4pyGyS0WDUT
I7CN7LZNGxjHd47Z
5add9cc3221ca69772a6f256036f72b14cc3652c53c15710511bdac7a533de5a
```

The result will be true if the signature is valid.

```
true
```

ANNOUNCING PBAAS TESTNET UPGRADE v0.9.1-2, OPTIONAL FOR MAINNET USERS -- MANDATORY FOR TESTNET USERS

v0.9.1-2 RESETS TESTNET -- THERE ARE NO MAINNET CHANGES

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.1-2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Testnet Featuring Verus PBaaS

The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

Verus PBaaS Features Live on v0.9.1-2 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.

- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.

- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.

- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.

- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).

- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.

- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:

    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.

- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.

- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

    - Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

    - Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals
```

```
OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals
```

```
OPTION_TOKEN = 32                // if set, this is a token, not a native
currency
```

```
OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

    "name": "PBaaSChain",

    "options": 264,

    "currencies": [
```

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

```
],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },
```

```
      {

        "networkaddress": "111.111.111.112:10000",

        "nodeidentity": "Node2@"

      }

    ],

    "gatewayconvertername": "Bridge",

    "gatewayconverterissuance": 1000000

  }
```


The bridge definition has overridable defaults


```
  {

    "currencies": [

      "VRSCTEST",
```

```
      "PBaaSChain",

      "USD"

    ],

    "initialcontributions": [

      380228.12033701,

      0,

      1000000

    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
```

```
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternamе":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/c0989e59ecee58bb1936d0cbe4f171a9b97b679730bccfae372e59575f8d9bff/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/0cbecd05a8dca2f73f959e35206b8d07711ac9c27e9c26afc2605ca315ef825b/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/c6f262bc03d0add6e8f593f455c661b8eafb7ad4d2ee94c13d2796226f4c4a58/detection
Windows:
https://www.virustotal.com/gui/file/386f6dfd78419d45412a94d7c1b293a0bbd12749cd9debf62e8196ea2cc0c762/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AYrcHQABQSBCgRc2BNPaD7R7lXD3BRfwV2dHQ7yyJ1qB/ZDX03Jk5VoHZdMG5c/V/j1rZCVfXWQm66mo
mTr4Ee05L+mfjf9l /Downloads/Verus-CLI-Linux-v0.9.1-2-x86_64/Verus-CLI-Linux-
v0.9.1-2-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AYrcHQABQSBCgRc2BNPaD7R7lXD3BRfwV2dHQ7yyJ1qB/ZDX03Jk5VoHZdMG5c/V/j1rZCVfXWQm66mo
mTr4Ee05L+mfjf9l
1f9e4144c48e0df53778c0a0af2dcbde57f95dd119c2348fbe3343fb4500da38
```

The result will be true if the signature is valid.

```
true
```

ANNOUNCING MAINNET AND PBAAS TESTNET UPGRADE v0.9.1-1

CRITICAL FOR NOTARIES, POOLS, EXCHANGES, MANDATORY FOR TESTNET USERS, HIGHLY RECOMMENDED FOR ALL MAINNET USERS

v0.9.1-1 includes testnet fixes for the Ethereum bridge launch with advanced features, addresses all known issues on testnet, and includes sync improvements for testnet and mainnet.

Testnet Featuring Verus PBaaS

The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.1, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Verus PBaaS Features Live on v0.9.1-1 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.

- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.

- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.

- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.

- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).

- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.

- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:

    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.

- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.

- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

    - Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

    - Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals
```

```
OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals
```

```
OPTION_TOKEN = 32                // if set, this is a token, not a native
currency
```

```
OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

    "name": "PBaaSChain",

    "options": 264,

    "currencies": [
```

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

```
],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },
```

```
      {

        "networkaddress": "111.111.111.112:10000",

        "nodeidentity": "Node2@"

      }

    ],

    "gatewayconvertername": "Bridge",

    "gatewayconverterissuance": 1000000

  }
```


The bridge definition has overridable defaults


```
  {

    "currencies": [

      "VRSCTEST",
```

```
      "PBaaSChain",

      "USD"

    ],

    "initialcontributions": [

      380228.12033701,

      0,

      1000000

    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
```

```
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternamе":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/cf645e9513dfcbd8e2e43eb330bb9090e91fd8f33a9521a86a75f90ff998ecc1/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/0ce97d7a308077827cea5dbfc37f48dc6d253ff5f0924db4d58abaf4068cd22e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/0088ad441a3bed5a134d1fbf5752f8ffed31bfd226bf5142e4c41ec02e2bb46f/detection
Windows:
https://www.virustotal.com/gui/file/fc110cfa3ab9eab71b7661628d9d0dc4fcfdb450801c41e81576a99e35244a5e/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AU+tHQABQR+/ObpwL0kRP/cIf4PvNMa/GBJa2f82LrGDGF8S45GHOlfZwHM7kL5X0Qst+1IVD8O1adW8
TeAlqZY6EIOh0Q5/ /Downloads/Verus-CLI-Linux-v0.9.1-1-x86_64/Verus-CLI-Linux-
v0.9.1-1-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AU+tHQABQR+/ObpwL0kRP/cIf4PvNMa/GBJa2f82LrGDGF8S45GHOlfZwHM7kL5X0Qst+1IVD8O1adW8
TeAlqZY6EIOh0Q5/
62819af68f302a39a02fe17ab3b917f4c51f6286e67dc3966718fe059299ff6c
```

The result will be true if the signature is valid.

```
true
```

ANNOUNCING NON-MANDATORY, RECOMMENDED FOR MAINNET, MANDATORY FOR TESTNET PBAAS UPGRADE v0.9.1

Testnet Featuring Verus PBaaS -- The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

v0.9.1 improves mainnet bootstrap and GUI backup/restore user experience but does not upgrade mainnet protocols and is not a required update for mainnet-only users. It is required to use the public testnet. IF YOU HAVE ALREADY USED TESTNET WITH PRIOR VERSIONS, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Verus PBaaS Features Live on v0.9.1 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.

- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.

- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.

- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.

- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).

- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.

- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:

    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.

- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.

- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

    - Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

    - Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs

- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

Currency Options

```
OPTION_FRACTIONAL = 1              // allows reserve conversion using base
calculations when set
```

```
OPTION_ID_ISSUANCE = 2             // clear is permissionless, if set, IDs
may only be created by controlling ID
```

```
OPTION_ID_STAKING = 4              // all IDs on chain stake equally,
rather than value-based staking
```

```
OPTION_ID_REFERRALS = 8            // if set, this chain supports
referrals
```

```
OPTION_ID_REFERRALREQUIRED = 16  // if set, this chain requires referrals
```

```
OPTION_TOKEN = 32                // if set, this is a token, not a native
currency
```

```
OPTION_SINGLECURRENCY = 64       // for PBaaS chains or gateways to
potentially restrict to single currency
```

```
OPTION_GATEWAY = 128              // if set, this routes external
currencies



OPTION_PBAAS = 256               // this is a PBaaS chain definition



OPTION_PBAAS_CONVERTER = 512     // this means that for a specific PBaaS
gateway, this is the default converter and will publish prices
```

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":32,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":33,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":20000'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
  {

    "name": "PBaaSChain",

    "options": 264,

    "currencies": [
```

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

```
],

"notaries": [

  "Notary1@",

  "Notary2@",

  "Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

  {

    "networkaddress": "111.111.111.111:10000",

    "nodeidentity": "Node1@"

  },
```

```
      {

        "networkaddress": "111.111.111.112:10000",

        "nodeidentity": "Node2@"

      }

    ],

    "gatewayconvertername": "Bridge",

    "gatewayconverterissuance": 1000000

  }
```


The bridge definition has overridable defaults


```
  {

    "currencies": [

      "VRSCTEST",
```

```
      "PBaaSChain",

      "USD"

    ],

    "initialcontributions": [

      380228.12033701,

      0,

      1000000

    ],

    "initialsupply": 3000000

  }
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":264,"currencies":["VRSCTEST"],"conversions":
```

```
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternamе":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/3c6ec8e701fdbfc966ae7e4606898aebd4f32538ef97ad32a94b723731becdf0/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/ed0d96a02e617a4d5aaa3a396dd28d9b2378e7377ef9236c31946bb4afea5a3e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/29289f0bb1c6a83b6a0a323ed867c7bb3857a33ca189397728f49ad513ce6641/detection
Windows:
https://www.virustotal.com/gui/file/55b8df070559fc7e4010a1b9466b4c7cce529c464eee17627f8409eaf61ac623/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ARh1HQABQSDoIbL907lmsx8IkKa0e3YT88VnW6KS47hnb5NbnegPSmVwVzx09t9/Y01Y1eJWMHMILnCg
rF8hdGSi/75TBfvk /Downloads/Verus-CLI-Linux-v0.9.1-x86_64/Verus-CLI-Linux-
v0.9.1-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ARh1HQABQSDoIbL907lmsx8IkKa0e3YT88VnW6KS47hnb5NbnegPSmVwVzx09t9/Y01Y1eJWMHMILnCg
rF8hdGSi/75TBfvk
ca05b85e8124d1c8f8a781616ef48119bca64f746715361e4e44e88b3830b911
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-3. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is recommended for testnet users to ensure the best experience with testnet.

This version fixes the following issues with v0.9.0-2

- Adds GUI support for seeing, making, and taking offers for currency or identity trades on the VerusID tab and when searching for identities.

- Fixes an issue where offers to buy or sell IDs were prematurely closed, even when expiry was set far in the future

- Fixes an issue with setidentitytimelock that caused it to return an error in most cases

- Ensures full rescanning of identities when using z_importwallet

New Command and API for Verus Vault - setidentitytimelock

v0.9.0-2 introduced a new command and RPC API, "setidentitytimelock". setidentitytimelock may be used to set a time unlock delay, set a lock until a specific block height, or unlock a delayed timelock on a VerusID under your control.

You can use setidentitytimelock in one of two ways:

```
verus setidentitytimelock '{"unlockatblock":1796500}' (returntx)
```

or

```
verus setidentitytimelock '{"setunlockdelay":1440}' (returntx)
```

or

```
verus setidentitytimelock '{"unlockatblock":0}' (returntx)
```

The first example will lock the ID until the block specified. If the block specified is in the past, the ID will not be locked.
The second example will set a 1440 block unlock delay, which will be needed to wait before spending funds controlled by the ID after unlocking the ID (about one day unlock time).
The third example will unlock an ID that has been locked with an unlock delay, and the unlock time will be set to the block height at which the ID was unlocked + the originally set unlock delay.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
   verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```json
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1
```

```
],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [
```

```
380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/a373bb52686ec4bae4f77843e8c806179b6e20c40f1f4855ab7f76c70f9e962a/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/c77976b43222ddecbea829cf5805d6f5d0b062b52f3c7112bde5e51c94d6a5e0/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/31aad050a7f726ed359598d506805882ca48b16d0015a1d156c7b41a786bfa3b/detection
Windows:
https://www.virustotal.com/gui/file/bddd820126a27512872d9dd8c966fcf253fe46777288b452ce22e8dcc1352ef0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AdJ2GwABQSCjCvnuaFl4Crj1vRK7s3gPAGa7u3OktJVrSY6NQoj002JK4IcK9xx0yQue/
LK0Kg7H6L+h42BTTZb9+zeLgGVs /Downloads/Verus-CLI-Linux-v0.9.0-3-x86_64/Verus-
CLI-Linux-v0.9.0-3-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AdJ2GwABQSCjCvnuaFl4Crj1vRK7s3gPAGa7u3OktJVrSY6NQoj002JK4IcK9xx0yQue/
LK0Kg7H6L+h42BTTZb9+zeLgGVs
fd58ff4f7f25a3eb2d48ce0b022c78162df66275c9f2257e071c5b97e9d94812
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-2. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is mandatory for testnet users to ensure reliable connection to the current testnet.

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER. IF YOU HAVE NOT UPDATED AND ATTEMPTED TO STAY CONNECT TO MAINNET, YOU MAY NEED TO BOOTSTRAP OR RESYNC YOUR NATIVE WALLET OR NODE AFTER UPGRADING.

New Command and API for Verus Vault - setidentitytimelock

v0.9.0-2 introduces a new command and RPC API, "setidentitytimelock". setidentitytimelock may be used to set a time unlock delay, set a lock until a specific block height, or unlock a delayed timelock on a VerusID under your control.

You can use setidentitytimelock in one of two ways:

```
verus setidentitytimelock '{"unlockatblock":1796500}' (returntx)
```

or

```
verus setidentitytimelock '{"setunlockdelay":1440}' (returntx)
```

or

```
verus setidentitytimelock '{"unlockatblock":0}' (returntx)
```

The first example will lock the ID until the block specified. If the block specified is in the past, the ID will not be locked.
The second example will set a 1440 block unlock delay, which will be needed to wait before spending funds controlled by the ID after unlocking the ID (about one day unlock time).
The third example will unlock an ID that has been locked with an unlock delay, and the unlock time will be set to the block height at which the ID was unlocked + the originally set unlock delay.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/94d81f9760a29d0407620a6dc0cd78af0518a34898a56e70b20224e99c4f2cb7/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/d84eca262fb1b4a44901eb6a334924cd1ab8520d479bb46b2d3b213b1b628d9e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/9e5a0b54b634b7dd4f389de51a67eacaa1c0317981b6aa05fad5e0f4c095e409/detection
Windows:
https://www.virustotal.com/gui/file/b0008b14b4273f76c23f0090b19d5dbbae08758cc78094c13b811e25ba4a51af/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQVqGwABQR/OeunkOwu/sRpFtHBBU33MD6DkvkFnM8vpFTBmNQF1m3eAtOyetX0LuTcUxjuc1KMX/
vj15Y54e0OSwzYWtLAA /Downloads/Verus-CLI-Linux-v0.9.0-2-x86_64/Verus-CLI-Linux-
v0.9.0-2-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AQVqGwABQR/OeunkOwu/sRpFtHBBU33MD6DkvkFnM8vpFTBmNQF1m3eAtOyetX0LuTcUxjuc1KMX/
vj15Y54e0OSwzYWtLAA
c814ca39dae00ca2152ab01ed3bd6b4a10e5d0b10320677f08686fb15c9bd126
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-1. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is mandatory for testnet users to ensure reliable connection to the current testnet.

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyoe with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/5b250aae5e5ae396a2ca74de8f59f3840098590a299c05cb26733e2d5a5b6cce/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/e8f4b9d83089e73d1f3e9e314c1d459f7fa3d28ee38cae6eaa3f17f2904be453/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/04c6d45b62873d4ff56103a07d85d0e0ffcf84416a434ce1c96e4e987132f88a/detection
Windows:
https://www.virustotal.com/gui/file/f9cc46ba66cd73424604fb7f596ff61f709527724d78252d283373e82c356ad4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AXY1GwABQR+zDdT4vg66nRnK/0lvyxWRsHUf0tg6mAzJSv7a5BVrpzCjxnKr3zOPhxtrxpaNOU5duU4+
69S/pqY7HSq4k6MJ /Downloads/Verus-CLI-Linux-v0.9.0-1-x86_64/Verus-CLI-Linux-
v0.9.0-1-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AXY1GwABQR+zDdT4vg66nRnK/0lvyxWRsHUf0tg6mAzJSv7a5BVrpzCjxnKr3zOPhxtrxpaNOU5duU4+
69S/pqY7HSq4k6MJ
d2e5fc890f646773624c55797d2dd38bab35a2839ad208ec0b668d34b24b27d3
```

The result will be true if the signature is valid.

```
true
```

Announcing MANDATORY Verus v0.9.0 Mainnet Release of Verus Vault and the on-chain NFT and ID Marketplace

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyoe with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/66f08fb597b6e68d7c5f6c93ac444cfbaec2d03535a31f6d86d6f164d8a1bd05/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/697656f6243a66d2ba40cbef3028edbeccbe5c9b8222c36617ea89ed94b5b996/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/7bbe91e83713e57c391df8ab25e9ce95d24ede9605fdfbcd72a7d140419e5657/detection
Windows:
https://www.virustotal.com/gui/file/cf2e044ef9a061dbee93f12fc06c26342bbce6ea7cfa93ebed656de3d4ce3b99/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AeoZGwABQR9Qyoon3B6wDze32NWnzFLRIl5mTH+OqmWNNxvAPDB+8CUTYBVcMAfiMOT87yGhfr8gn4Kr
QmpTe1Mllm0yjQRu /Downloads/Verus-CLI-Linux-v0.9.0-x86_64/Verus-CLI-Linux-
v0.9.0-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AeoZGwABQR9Qyoon3B6wDze32NWnzFLRIl5mTH+OqmWNNxvAPDB+8CUTYBVcMAfiMOT87yGhfr8gn4Kr
QmpTe1Mllm0yjQRu
2c29c3af72e33117e28ac79b060abe5d87d892c4ed7204b40125c674afd2403f
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-5 UPDATE - ONLY REQUIRED FOR THOSE CONTINUING TO HELP GET TESTNET READY FOR MAINNET

We want to thank the Verus Community for bearing with the number of updates we've had to get the bridge to Ethereum working smoothly. This release both fixes the remaining known bridge issues and adds security to ensure that you don't send to an unspendable address on any network by using the wrong format for a destination.

The good news is that throughout the whole process of this first bridge release on testnet, we've been able to update through the issues we've seen without a reset and have not lost a single satoshi of test funds (which is not to say it is fully hardened quite yet). At this point, we hope and believe that this is the last update in this round of testnet, which will allow us to get back to focusing on the mainnet release and not asking those testing to update until we are very close to a mainnet release.

Once more, we will ask that you shut down the old version now, and we will restart the notarizations and transaction flow between Ethereum and Verus at 5 AM UTC time.

Thank you to everyone who helped with all the testing and exposing of issues that enabled us to make maybe even inconveniently quick progress.

If you miss the fork for this version at the above time you can get back on the correct chain by stopping Verus and deleting deleting everything other than your wallet.dat file in your data directory at:
- Linux: `~/.Komodo/vrsctest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest` or `%AppData%\Komodo\vrsctest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/ff594a0d654578f460d0668c745cbb5440c888fe506141c8d6be636aae2d89ba/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/ba5d35f051c235f7340b9e4f020a7dbdfcea714be8d906bddb45d4195b84a21e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/cd2df860d9a2813d069a94de43af89950c4553d18b513392346b23418a7cb490/detection
Windows:
https://www.virustotal.com/gui/file/af461a1ee81a3dd3e6bd25cb38213d0ce16f23b690a85a71ea831e2d87af48cb/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AaObGgABQR8BNRwDtlS1dqFyJEIEJMxKmQijlwW5KWZVIrPkIEjWCjzFZak9ZBIWyrFObHHPNrdE72mV
CUblg1Ap4fzMfk1D /Downloads/Verus-CLI-Linux-v0.8.0-5-x86_64/Verus-CLI-Linux-
v0.8.0-5-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AaObGgABQR8BNRwDtlS1dqFyJEIEJMxKmQijlwW5KWZVIrPkIEjWCjzFZak9ZBIWyrFObHHPNrdE72mV
CUblg1Ap4fzMfk1D
157349879796973a3b1dbbfd446da72126484393942cc1c87615620d1ac2ad8b
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-4 UPDATE.

This release has no mainnet changes, resolves issues that people using testnet have reported with the Ethereum/Rinkeby bridge, and is mandatory for those who wish to continue using testnet after transactions between the Verus and Rinkeby testnets resume flowing.

If you have pending transactions on the bridge, some requested conversions may come back as refunded, and you should receive all amounts expected, less fees. We will not reset the testnet, but within an hour after this availability, the network will resume transaction imports/exports to and from Rinkeby, and anyone who continues running the old version may need to resync to testnet from scratch to continue. If you have upgraded to v0.8.0-4 or do so before connecting again to testnet, you should only see improvement without disruption.

I want to thank everyone who participated already in using this testnet and bridge release. I hope we can keep that level of participation or more up, as it has been extremely helpful. Because of all that's happened on this testnet to date, we were able to make a lot of progress in finding and resolving the following testnet issues in this release:

1. Enable support for much larger proofs than was previously possible from systems that either have large proofs required or significantly variable length proofs. Having this capability in the daemon will pave the way for more types of cross-chain proving in the future as well.

2. Fix failed refunds of conversion transactions attempted after a fractional currency launches, but before it's preconversion phase is complete. If you sent transactions during this time, you may have stuck refunds that should flow after this upgrade.

3. Ensure that currencies are marked as "launch complete" immediately after all preconversions have been processed, enabling normal conversions in all directions from that point forward.

4. Fix an issue that left some import notarizations incomplete and caused cross-chain transfers to wait longer than expected after some imports.

5. Improve and validate hardening.

If you miss the fork for this version at the above time you can get back on the correct chain by stopping Verus and deleting deleting everything other than your wallet.dat file in your data directory at:
- Linux: `~/.Komodo/vrsctest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest` or `%AppData%\Komodo\vrsctest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":100,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/c0662bb8a69114058d57a8f096170b3787e29b8ce7e9305845b005259371556a/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/1b8c8d898253ba0d665db73cbbcc05d346e2cc0160d2d99184bd4ce45167bdd3/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/d0d3a63cf95f0a5b350c14d5e4b3940ec2f1db40f240209dad18a212cfea8644/detection
Windows:
https://www.virustotal.com/gui/file/61fb27dda606908962f20cc9b20416e6bc066afbca81647e74a56b1395d8e645/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AfKUGgABQSA/OfjAZGEy1HhA1iTKRDvPLV+cXXbVfn+3IClAtsqIHl4Q6IuIClHsynG4FnE5ttW8rxuS
NG5EXezmoMRkzxXH /Downloads/Verus-CLI-Linux-v0.8.0-4-x86_64/Verus-CLI-Linux-
v0.8.0-4-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AfKUGgABQSA/OfjAZGEy1HhA1iTKRDvPLV+cXXbVfn+3IClAtsqIHl4Q6IuIClHsynG4FnE5ttW8rxuS
NG5EXezmoMRkzxXH
0ca20fcf2480be882d17bc77ad693b7a4e918502bfc7c67592dfb0c32ba26199
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-3 UPDATE.

THIS UPDATE IS ONLY NEEDED FOR THOSE COMMUNITY MEMBERS USING TESTNET. THERE ARE NO MAINNET CHANGES IN THIS VERSION.

You will need this update to stay connected to testnet once we restart bridge traffic in about an hour. If you do stay connected on the old version and fork away from testnet, you should still be able to start with a reindex and rescan after upgrade.

v0.8.0 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.

2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.

3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.

4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.

5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.

6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.

7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
   verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```json
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1
```

```
],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [
```

```
380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/4bbaf0dbd73a748226933ccb1b101fa3874f3d5549de4fd334735d3ed0ebe50e/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/62a353c6c39512a2779629dcefc90c2d64104ad97f0af798d5269ab283e00ce7/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/1344d8d98b5e42aa1dff1dc380a8c8f20c555d296967078c9c18ae560c2cc0fc/detection
Windows:
https://www.virustotal.com/gui/file/c4d5998b8dd133f029387c7eff92db883e75c6024a8e868be3d2b6feca8b98a5/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AVZ7GgABQR/RKbDtXeDnzBNFkmE9eWv9WRLchaCxUchK2VOmtxqubmqobR9g3SrX0J+OSrVlsElix6RI
xCxyHAGjwQ+zsvWv /Downloads/Verus-CLI-Linux-v0.8.0-3-x86_64/Verus-CLI-Linux-
v0.8.0-3-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AVZ7GgABQR/RKbDtXeDnzBNFkmE9eWv9WRLchaCxUchK2VOmtxqubmqobR9g3SrX0J+OSrVlsElix6RI
xCxyHAGjwQ+zsvWv
dcaa6b06eef134429e7c077396bc5957228985e8ab70d2b34058d0da02721243
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-2 UPDATE

This daemon release includes a fix that simplifies some bridge transactions, and is the version you should run on testnet when the Ethereum bridge does launch for a good experience.

v0.8.0 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.

2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.

3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology

is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.

4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.

5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.

6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.

7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
   verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/5f76dafe4c5569ae6ee1ea9ebf57a348c6f0536f5c8e2269c88788d55a5eeb10/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/2bb6e95451ab19b4480422652cc4b53ea23c000a8d5c1f21efba6423c8558492/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/17029a232661259b9f0ee8daf5a9cd99e7f5899130c657fc96b103801fbb3d32/detection
Windows:
https://www.virustotal.com/gui/file/02f23519de5afdfa6afbcb202321f58d8e277afaa1be08781cb1130e86abf48b/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQh2GgABQR+PYOQxgJtQzB8wwbU+VvPvLApcEfAmvJ+jI78h/qPVTnU0uJrpy4Fvc7yGCWHT1Awh3NSG
RFbYQAVQB7u+5XWi /Downloads/Verus-CLI-Linux-v0.8.0-2-x86_64/Verus-CLI-Linux-
v0.8.0-2-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AQh2GgABQR+PYOQxgJtQzB8wwbU+VvPvLApcEfAmvJ+jI78h/qPVTnU0uJrpy4Fvc7yGCWHT1Awh3NSG
RFbYQAVQB7u+5XWi
414f4f2d090d329602584ca5c665ecf04ab00570ce997f78bd5b50e55e3dc977
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-1 UPDATE

This release includes a testnet fix for the Ethereum bridge and is required for those who want to continue using the current testnet.

v0.8.0 and v0.8.0-1 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.

2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.

3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.

4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.

5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.

6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.

7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/d4ddf18646a1b3e3af8e8b8b2bd77ec8db0394b4be1d5c074d13816a67e171e1/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/f4656745fe3b80f67abbe4180b39e3686f3ec28ec75846cf318d8cdabfdc1daa/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/ac1b3e41767fd8c3af6eef780c73bf43097e00baa0884b95097a296c3030791f/detection
Windows:
https://www.virustotal.com/gui/file/df371574c420534fc91344cca873f464cab8f5ff5a67f8741aadfe764c676434/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AeduGgABQSCt7p7H5gtjUkrIAAk5SX9ss9xXz5Z+Qikf61vRPnWGSyX6tLaO62BMNTKDE7sJVA9/
Bp89yR1fZZ6h04ARIyYT /Downloads/Verus-CLI-Linux-v0.8.0-1-x86_64/Verus-CLI-Linux-
v0.8.0-1-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AeduGgABQSCt7p7H5gtjUkrIAAk5SX9ss9xXz5Z+Qikf61vRPnWGSyX6tLaO62BMNTKDE7sJVA9/
Bp89yR1fZZ6h04ARIyYT
998170ee70cca3116bc100ef1b972fd919a39b069ebd1dd5a6e1df30f9abacc1
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY, RECOMMENDED v0.8.0 UPDATE.

This release is mandatory for those who want to use the testnet, which will be reset as of v0.8.0.

Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.

2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.

3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.

4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.

5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.

6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.

7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/43fefb235a9b17f36f20f2746094f21459a89aa4c2c7b48f6eb5c7d7855252c1/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/616b02da7e59dad21892298b3e0e7131196f4bc2e328ab3fb69d37b3cdc40ebd/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/a87e6af87dd2359ae01bff7be5219321d1883cda140a250d99bfc30b62d3a855/detection
Windows:
https://www.virustotal.com/gui/file/1b4fe15fe9185a298188d39733e271f4715e4090bbb1432ef7bc5332ff7bf823/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
Aa5ZGgABQR8UGSp8PtL3P07qUBibDAn0pyeOunSohgRFFFljIBs13zXe+9SZlZB8Tm7Xd1gF1ikjseaI
cgU1gXG0sG183uwC /Downloads/Verus-CLI-Linux-v0.8.0-x86_64/Verus-CLI-Linux-
v0.8.0-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
Aa5ZGgABQR8UGSp8PtL3P07qUBibDAn0pyeOunSohgRFFFljIBs13zXe+9SZlZB8Tm7Xd1gF1ikjseaI
cgU1gXG0sG183uwC
0f87efe340bb97679e58eca045ff996483ba3bc579250672a74090f7cf282580
```

The result will be true if the signature is valid.

```
true
```

ANNOUNCING HIGHLY RECOMMENDED NON-MANDATORY, CRITICAL FOR NOTARIES AND POOLS UPDATE, v0.7.4.

Version v0.7.4 can improve a node’s synchronization and this version also upgrades and resets the Verus testnet, preparing it for the testnet launch of the Ethereum/Rinkeby bridge and bringing Verus PBaaS multi-chain, multi-currency, Verus Vault ID timelocking, Verus MEV-resistant DeFi, and the Ethereum bridge much closer to a mainnet ready state.

This release fixes all known functionality issues reported from the previous testnet including some related to currency and blockchain launches, as well as being incorrectly able to reduce the timelock period on a Verus Vault locked ID. It also includes the ability to export an ID from one chain to another, using the “exportid”:true parameter in sendcurrency, enabling cross-chain use of VerusIDs. With this release, we will be resetting the Verus testnet. To use the new testnet, you must make sure to delete the vrsctest and .verustest/pbaas or VerusTest\pbaas folders beforehand.

In addition to being prepared for the Ethereum bridge launch, this release adds some new testnet capabilities, including the ability to send to IDs cross chain and export IDs from chain to chain using the `“exportid”:true` parameter with `sendcurrency`. IDs only need to be exported once and can then be used and managed directly as a forked ID on the destination chain.

Exporting an ID to a PBaaS chain

```
   verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/0e935960b0886275c13e43a65771e4f5e89f341a450eb6f374576d26b244bf53/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/2a783bdc3367b8475d97d9da9d1b8c3510e9ab172102df39626cfd95e08d92a6/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/489fb285e70e0e58fa1f52453a16b194a743a6d8adb5f38436680c32e681e8db/detection
Windows:
https://www.virustotal.com/gui/file/043112c9d81dfdc57120bfd8d18516adbd836c700ec60e51455dac0a615bf276/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AdZEGQABQSCMoIi7MwgJwB2fXqry1Q2jMu8rIQUiRq+zRezdyhomTGwO1aVZN048dPAJR7dTIDUoA+os
Y2mx1Kfw33S8D5R7 /Downloads/Verus-CLI-Linux-v0.7.4-x86_64/Verus-CLI-Linux-
v0.7.4-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AdZEGQABQSCMoIi7MwgJwB2fXqry1Q2jMu8rIQUiRq+zRezdyhomTGwO1aVZN048dPAJR7dTIDUoA+os
Y2mx1Kfw33S8D5R7
2b7fcb26fc38f34e1fd77e581716c1cd8b4eee1f0f01627dfd2ef56170a8e2f4
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY, RECOMMENDED UPDATE FOCUSED ON NETWORK CONNECTIVITY AND ROBUSTNESS IMPROVEMENTS v 0.7.3-10.

v0.7.3-10 has very minimal updates targeted at networking robustness - faster response to notary inconsistencies and a check for detecting a corrupt index that if present, might otherwise prevent a node from staying synced.

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
   verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```json
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1
```

```
],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [
```

```
380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/3ac4a3b7f2c27c41541ca385d1e03f77f45996ddb2363acd21802a7b7d21adb1/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/f623b2042943830f1be3e6002dcdf6592225c357c3837d850e7f60a1311d3059/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/390266baf97f9ad3c8c1286479e42675ae4e0b065b581fc5234ed3c104a96002/detection
Windows:
https://www.virustotal.com/gui/file/4632bb21603166253e8fcc2910e2daa7a6428b11c5203f90d1debbe210b3c9a0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ARJwGAABQSAEzdWu0ba5tX9dSzAPIJsuuYhh/+Fd4F3m0uPsQHLCWCr6PGjzCB+aC5uBtVxtheSL1ATW
W6nDv6/OwyEDtK4V /Downloads/Verus-CLI-Linux-v0.7.3-10-x86_64/Verus-CLI-Linux-
v0.7.3-10-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ARJwGAABQSAEzdWu0ba5tX9dSzAPIJsuuYhh/+Fd4F3m0uPsQHLCWCr6PGjzCB+aC5uBtVxtheSL1ATW
W6nDv6/OwyEDtK4V
f99dfbd27d8b97eb14861d37371858c545a293e56ec8dd742ffa4f2365966af9
```

The result will be true if the signature is valid.

```
true
```

Announcing immediate availability of NON-MANDATORY, RECOMMENDED ESPECIALLY FOR NOTARIES, HIGH PEER COUNT PUBLIC NODES, AND POOLS, v0.7.3-9 Verus CLI.

In addition to more immediate and effective bad node banning behavior, v0.7.3-9 includes a fix for signrawtransaction that could sometimes result in failure of countersigning a multisig, ID-based raw transaction.

While this release does include network robustness and a multisig fix, we were also able to upgrade Verus multisig capabilities to an ease of use and level of capability beyond other networks we are aware of today. In this release, it is now possible, on both mainnet and testnet, to issue any “sendcurrency” command with the source being a multisig ID, for which you do not have all the keys present in your wallet. If you do, there will be a return error in operation status, from both GUI or CLI, and in that error will be the partially signed transaction, which can be used by applications or with copy and paste to send to other signers with partial authority over the source ID. On mainnet, destinations for such a transaction can include IDs, private addresses with memos to a z-address or ID:private, and transparent addresses. On testnet, such transactions can include all that is possible on mainnet as well as multiple currencies, currency conversions, and cross-chain sends.

To our knowledge, this release raises the industry standard for multisig capability and its support on public blockchain protocols to a new level, even before considering the multisig revocation and recovery capabilities.

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
   verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
   verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
   verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
   verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```json
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1
```

```
],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [
```

```
380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/9b1bf2f784c08ae1d823e35591fbce52f82368e61dd7d3bccabaacc0d1495ab6/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/cf672a3d32b7578cf1f44173d7fc85f755029c738eb0f7a0e17351fba5d8a477/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/ed83e96889c39e1cd01207e917cc470133ff2dc1f23c95c8c5daf98c05d861e6/detection
Windows:
https://www.virustotal.com/gui/file/f3c83c88cb5b8633c54c2b5ad3fcfe1d897fe44fdd72d37478e8816814825d06/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ARg+GAABQR/kRkEtGRrU8w0LW6gFeGNrMwEpWNIbB434GJWxYFuJHFX9pK5OthceonYqPe9WVjbZX95I
pyONnzaYVndL5hfH /Downloads/Verus-CLI-Linux-v0.7.3-9-x86_64/Verus-CLI-Linux-
v0.7.3-9-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ARg+GAABQR/kRkEtGRrU8w0LW6gFeGNrMwEpWNIbB434GJWxYFuJHFX9pK5OthceonYqPe9WVjbZX95I
pyONnzaYVndL5hfH
444d2c231e9d76bcff47c0f3da6822aab7c39567a2c313642675a35fcd484f7f
```

The result will be true if the signature is valid.

```
true
```

Announcing MANDATORY UPDATE v0.7.3-8, which will re-enable full Verus Proof of Power, including staking, at block 1576200, occuring on Thursday at approximately 5:00PM UTC time. YOU MUST UPDATE BEFORE THEN TO FOLLOW THE MAIN CHAIN.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can

now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/a85b5386f0c59ec851dc12f3acd789575619d9b31345b6e871f4c9aa98fcc13d/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/be2d22663f0c84ca37bd782870fd578097375d43f735ee7389ee5ded35743dc6/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/143ca9ba408ae89027a70dd0ac3770554b1937368c5e24f04233d36b5995d651/detection
Windows:
https://www.virustotal.com/gui/file/1a286ac9e66424d5eb33f35c2a4c3e8303a15ea7c1dc9ae808991b7456c100dd/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AW79FwABQR/Il7BalzNJsvWF64L7iXUQP9nQ1nZ60bZe9HQ5yxTH2ybQlnSSBaJd2C3iJSqemuCl36Zc
12GEwHPooyUP+EWm /Downloads/Verus-CLI-Linux-v0.7.3-8-x86_64/Verus-CLI-Linux-
v0.7.3-8-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AW79FwABQR/Il7BalzNJsvWF64L7iXUQP9nQ1nZ60bZe9HQ5yxTH2ybQlnSSBaJd2C3iJSqemuCl36Zc
12GEwHPooyUP+EWm
7b77f2eccdd85ae2a834ea5ddfdcb70e275d9d78b9eb435cc4a1127aecf57fab
```

The result will be true if the signature is valid.

```
true
```

Announcing MANDATORY UPGRADE v0.7.3-7. This release fixes the bug that was exploited by over the last day which stalled the network. You must upgrade to this release. If you shut down prior to block 15679999 you should be able to run this release without any additional steps. If your wallet continued to run after block 15679999 (or is still running) you will need to bootstrap or otherwise re-sync.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/172058d2b93de9bd91ff26b5c6d38828fd55c44a61f0d7439247acb800042197/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/b5f1c8c300a978936e4b555207e4d0cc371a47773affa459b4c0a0603d105d1e/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/97f852a89ed18f000ce85485a2af5a1e4b49115264666dca0bc8b1993374dd36/detection
Windows:
https://www.virustotal.com/gui/file/ff391d62965067a42994b41a84e16f95239a46cf0bade52efefd2a5c95e2a194/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AUX2FwABQR//3tYmE7EmZs5gVUIQCLdy8VI4D1/MBM53bRLpq3C2cXLScU4bdJek6orr9VQE5vPGnuR5
68r5uM2cVpBjzyGB /Downloads/Verus-CLI-Linux-v0.7.3-7-x86_64/Verus-CLI-Linux-
v0.7.3-7-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AUX2FwABQR//3tYmE7EmZs5gVUIQCLdy8VI4D1/MBM53bRLpq3C2cXLScU4bdJek6orr9VQE5vPGnuR5
68r5uM2cVpBjzyGB
b2edf4d8501851b655e62fe4c188589f9602d595837b1dec74825de4911022bc
```

The result will be true if the signature is valid.

```
true
```

Announcing MANDATORY UPGRADE v0.7.3-6, REQUIRED FOR KOMODO NOTARY SEASON 5. Notaries, pools, and exchanges should upgrade to this version as soon as possible, before notary changes take effect on block 1562500. Failing to update will not prevent connection to the network, but updating in a timely manner will ensure a stronger, notarized connection to the Verus blockchain.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/4bba9bcf3ac9f6c497af79debc4b2c022dbaef166b035b1848149c0e9871ddff/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/f8589343298ec7bfbaf94060addeb7c37e538ed7a70d987f567936f54bcb31c0/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/bca3b56057bb70b6c22040925d512ccb7e56b2b3e07bdb8120633f8cd1d51f5d/detection
Windows:
https://www.virustotal.com/gui/file/7ed190ffeea201a864739214dc6fc0369408a420f020d59a43035d092ed177fb/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
Ab7DFwABQSCAskYMLjJd/Ej48rkhX4DPyAizzGXksKD/0Hh2JWau3hgKLsaUBx6dqlbAf4RBdfENtRdP
XLVcf66YOhH1pi05 /Downloads/Verus-CLI-Linux-v0.7.3-6-x86_64/Verus-CLI-Linux-
v0.7.3-6-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
Ab7DFwABQSCAskYMLjJd/Ej48rkhX4DPyAizzGXksKD/0Hh2JWau3hgKLsaUBx6dqlbAf4RBdfENtRdP
XLVcf66YOhH1pi05
00a651a40cdd453614abac914f793f18284f1738deee67ac332310f72c479b64
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY GUI upgrade v0.7.3-5, required to continue participating in the PBaaS enabled testnet. v0.7.3-5 resolves an ID wallet sync issue for mainnet that could occur when recovering a revoked ID, causing the wallet to double count some amounts. This issue could not result in either loss of funds nor a security issue. This release also resolves all known testnet issues on the current testnet and does not require a testnet reset.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/gui/file/f172e3e2bc2605e8d9ce8edfeca1b3946ba07fc605ee9f781d4dcae6d9a6a007/detection
Linux-x86-64:
https://www.virustotal.com/gui/file/8ae30dcd2ae5be047f9575188f53e34bf348691e908a4b3ba99c75371fd94bbe/detection
Linux-ARM64:
https://www.virustotal.com/gui/file/669d415ffeec29253b744a12bac8498aa8ef64ec3ee7dcccb78718699ba6909a/detection
Windows:
https://www.virustotal.com/gui/file/084a202bdecadd0579924120cfc39b1de6eec461a50b7cf1ec0c7edb2004ad87/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AUo/FwABQR/8Mse5lnb7stZDIBtVu6/Wy3NyvKNKpyZv/2chp7bm1g9umtlqqpv6Zd3D8xMtLHjiOZpX
gam8NpvR9g1WqvFO /Downloads/Verus-CLI-Linux-v0.7.3-5-x86_64/Verus-CLI-Linux-
v0.7.3-5-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AUo/FwABQR/8Mse5lnb7stZDIBtVu6/Wy3NyvKNKpyZv/2chp7bm1g9umtlqqpv6Zd3D8xMtLHjiOZpX
gam8NpvR9g1WqvFO
26b1c75156b4795505afc1236c1817bfa1dc15497d7ae952709b7345dd08659e
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY CLI upgrade v0.7.3-4, required for to continue participating in the PBaaS enabled testnet. Mainnet features are effectively unchanged from v0.7.3-3

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for

your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```json
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1
```

```
],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",
```

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

```
],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [
```

```
380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}
```

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
ertername":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to

the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/178eabac12e068ac214b7a4c5e6313abff42cdb8b5314583b1a36521d1e14fd4/detection
Linux-x86-64:
https://www.virustotal.com/#/file/ec422d6e0354a7ff62b785ba1381eef11da5c7d9eaf18e39838c6a69e206e5e8/detection
Linux-ARM64:
https://www.virustotal.com/#/file/d23e728d52bea80bc781108497d013daddd356af3968ec367eff075486b96ed9/detection
Windows:
https://www.virustotal.com/#/file/fbcf6ad4f09c33962fe9d3677a582cc66587dc8b3cd19fe9e523a075004ae240/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AUo/FwABQR/8Mse5lnb7stZDIBtVu6/Wy3NyvKNKpyZv/2chp7bm1g9umtlqqpv6Zd3D8xMtLHjiOZpX
gam8NpvR9g1WqvFO /Downloads/Verus-CLI-Linux-v0.7.3-4-x86_64/Verus-CLI-Linux-
v0.7.3-4-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AUo/FwABQR/8Mse5lnb7stZDIBtVu6/Wy3NyvKNKpyZv/2chp7bm1g9umtlqqpv6Zd3D8xMtLHjiOZpX
gam8NpvR9g1WqvFO
26b1c75156b4795505afc1236c1817bfa1dc15497d7ae952709b7345dd08659e
```

The result will be true if the signature is valid.

```
true
```

Mainnet Updates

v0.7.3-3 is a CRITICAL update with fixes for potential, targeted DOS attacks that do not affect overall network security, but should be considered mandatory for exchanges, pools, and businesses using the Verus network. Previous 0.7.3-x releases should no longer be used and should be replaced by this version.

While this release includes testnet functionality for Verus PBaaS, described below. This is a mainnet focused release that does not fix any issues on testnet. Testnet issues will be address in upcoming optional releases.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via

cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```json

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

```
  verus -chain=VRSCTEST sendcurrency "*Z"
'[{"currency":"vrsctest","address":"alice@:private","amount":10},
{"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"exportto":"Bridge.PBaaSChain"}]'
```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"PBaaSChain","exportto":"Bridge.PBaaSChain","via":"Bridge.PBaa
SChain"}]'
```

- **Converting PBaaSChain to VRSCTEST:**

```
verus -chain=PBaaSChain sendcurrency "*"
'[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
"convertto":"VRSCTEST","exportto":"VRSCTEST","via":"Bridge.PBaaSChain"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Defining a PBaaS blockchain

```
{

"name": "PBaaSChain",
```

"options": 330,

"currencies": [

"VRSCTEST"

],

"conversions": [

1

],

"eras": [

{

"reward": 1200000000,

"decay": 0,

"halving": 0,

"eraend": 0

}

],

"notaries": [

"Notary1@",

"Notary2@",

"Notary3@",

],

"minnotariesconfirm": 2,

"nodes": [

{

"networkaddress": "111.111.111.111:10000",

```
"nodeidentity": "Node1@"

},

{

  "networkaddress": "111.111.111.112:10000",

"nodeidentity": "Node2@"

}

],

"gatewayconvertername": "Bridge",

"gatewayconverterissuance": 1000000

}
```

The bridge definition has overridable defaults

```
{
```

"currencies": [

"VRSCTEST",

"PBaaSChain",

"USD"

],

"initialcontributions": [

380228.12033701,

0,

1000000

],

"initialsupply": 3000000

}

Now pass those definitions to `definecurrency`

```
./verus -chain=vrsctest definecurrency
'{"name":"PBaaSChain","options":330,"currencies":["VRSCTEST"],"conversions":
[1],"eras":[{"reward":1200000000,"decay":0,"halving":0,"eraend":0}],"notaries":
["Notary1@","Notary2@","Notary3@"],"minnotariesconfirm":2,"nodes":
[{"networkaddress":"111.111.111.111:10000","nodeidentity":"Node1@"},
{"networkaddress":"111.111.111.112:10000","nodeidentity":"Node2@"}],"gatewayconv
erternname":"Bridge","gatewayconverterissuance":1000000}' '{"currencies":
["VRSCTEST","PBaaSChain","USD"],"initialcontributions":
[371747.20398827,0,1000000],"initialsupply":3000000}'
```

Limitations in multi-currency to be aware of:

1. (GUI and CLI) You will not be able to make a currency from an ID that has a properly encoded i-address as its actual name, not its calculated ID. Generally, the advice is “don’t do that”. Using an i-address when referring to a currency will only be interpreted as referring to the currency or identity which has that i-address calculated based on its name registration. As a general rule, making an ID with an i-address as its name is not prevented by the protocol, but will cause problems whenever an i-address or name may be used (many cases) and will not be supported for creating currencies. To reduce any potential for user confusion, even though the naming system provides will also be expanding the set of characters that will not be allowed for currency registrations.

2. (GUI) If a currency is supported in the Verus Desktop already, such as BTC or ETH, you will not be able to use those same named currencies as a PBaaS chain. This limitation is considered errata for this testnet release and will not be a limitation before mainnet availability.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/vrsctest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/vrsctest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a89c74194eca46de3b6dc5fe4f996a8d4b2dc67e523fb8923f28d635fb4348fc/detection
Linux-x86-64:
https://www.virustotal.com/#/file/37facf48634f61639248801a0862b751cde571701ad95d20dea438860e4877ad/detection
Linux-ARM64:
https://www.virustotal.com/#/file/6d8468ac74996f52068e03a38d5c98f58e80fbf034c18a8bdb9eb5ab60782712/detection
Windows:
https://www.virustotal.com/#/file/6365be1b0af8b9d924111b4c014b2653c338e4e8717c47056b599f68e81a2931/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AakaFwABQR+eYI49GX0y/z1x0Abr375LlHTD01ADTTBfA5D9uVhGvTtILdO+DwOmnWgWw6twtNQ3RBOs
+jVHkVe04grz+NAO /Downloads/Verus-CLI-Linux-v0.7.3-3-x86_64/Verus-CLI-Linux-
v0.7.3-3-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AakaFwABQR+eYI49GX0y/z1x0Abr375LlHTD01ADTTBfA5D9uVhGvTtILdO+DwOmnWgWw6twtNQ3RBOs
+jVHkVe04grz+NAO
2db05535dbc92292e18aab2e2d0a057322fcca42e7ee106ef16e267063201765
```

The result will be true if the signature is valid.

```
true
```

Announcing MANDATORY CLI upgrade v0.7.2-8 - THIS UPDATE IS CONSIDERED MANDATORY - ALL NODE OPERATORS SHOULD UPDATE AS SOON AS POSSIBLE
This release includes the following improvements and updates:
- Critical updates
- Fixed fetch-bootstrap temp file cleanup on Windows.
- Fixed fetch-bootstrap checksum verification on MacOS

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could

immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```

```

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**
```

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/1b0f4d17cd324f2beb9618b4b9d1162996b2c63b6986d0ae9004abd65aa0a0cf/detection
Linux-x86-64:
https://www.virustotal.com/#/file/c42ac46640370876e8f31f7a5c02543715c55060380400572facd145c46d010f/detection
Linux-ARM64:
https://www.virustotal.com/#/file/c93b0945ff66656d1f336aa329139ae695c421145e826d3b63434b4cd7f78910/detection
Windows:
https://www.virustotal.com/#/file/cf644a10e6cd3a0246380dc275289679122581bbb47dd4c3ac9e873ce112a97a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AbRHFQABQSBEyhS6Ohi6c4XfZfcM2xvlBkdH3CVtRMBiKgVaX6Z1SShf7c27DiBGveW60y0UQ455O5oZ
edXxvBybddEWIFi/ /Downloads/Verus-CLI-Linux-v0.7.2-8-x86_64/Verus-CLI-Linux-
v0.7.2-8-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AbRHFQABQSBEyhS6Ohi6c4XfZfcM2xvlBkdH3CVtRMBiKgVaX6Z1SShf7c27DiBGveW60y0UQ455O5oZ
edXxvBybddEWIFi/
727f6f7e2055adffde5928e17a50ffb99ec5ba45e8bc6cc543cbcdd7d6cbd15a
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY CLI upgrade v0.7.2-7, with fixes necessary to continue participating in the current testnet. Mainnet features are effectively unchanged from v0.7.2-6.

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the

entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
```

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

```
```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/23aa86c409d0e625b1323db2ea1cfc70a35a3798eeb07ff34644dc85df86a526/detection
Linux-x86-64:
https://www.virustotal.com/#/file/0b41b7d91fcbdd37aeb9d5b233e346cb47ca6f25f6b267745b68696eddf9eda4/detection
Linux-ARM64:
https://www.virustotal.com/#/file/5d009b9d7e88669eb92374a82b05d8bf842214e113476a7244002d79aeb7819f/detection
Windows:
https://www.virustotal.com/#/file/29e06e25f887311e8fcbfd597406df19280f75e1aa2d5bff4516e4cd4ced3456/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AeLPFAABQR+s98mnsOWxeaCcqPQ/Fc8d5cZh8QJtbz/ZGXqAtlKtyx1fV2HX2wO755f3O6qeT+Oyvk8S
AK2bTA+rFOVsUQAk /Downloads/Verus-CLI-Linux-v0.7.2-7-amd64/Verus-CLI-Linux-
v0.7.2-7-x86_64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AeLPFAABQR+s98mnsOWxeaCcqPQ/Fc8d5cZh8QJtbz/ZGXqAtlKtyx1fV2HX2wO755f3O6qeT+Oyvk8S
AK2bTA+rFOVsUQAk
d24732b7d81eabdb15ea8bb775c6d833117f934694628a2e8c3c09fb7d8fa601
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY CLI upgrade v0.7.2-6, with primarily testnet improvements. Mainnet features are effectively unchanged from v0.7.2-4, with only "z" *added as a new, supported wildcard to the* `z_getbalance` *API.*
This release includes the following improvements and updates for testnet:
- Add `getcurrencybalance` *API to get multi-currency balances on transparent/private addresses and identities.*
- Optimize multi-currency operators and UTXO selection
- Display unlocked balance in `getwalletinfo`
- Add support for "z" wildcard in `z_getbalance`

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is

simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command

to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/b9f6e82a0e31dae0fe109023812d47a14f5b906d3ae3bbd1c0384f527c039a39/detection
Linux-AMD64:
https://www.virustotal.com/#/file/c7dab849dae13ec1a4af896724291c268972df961391dbe00beacd9517ded108/detection
Linux-ARM64:
https://www.virustotal.com/#/file/984889d97a3c15566997e7a125cedde3dcb221ff7c259bc060caa9246f90f8cd/detection
Windows:
https://www.virustotal.com/#/file/59e80853b669da2ee4c397c280af56d9bb68a159ea15cfaa386cdb1cafeb522d/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AV6kEwABQR85s2Rs3BjLBikfd0gDQ3CeUxJwRDzx0Y+DocPn5SzWdxq6W45PVDVDFBRPUzKY48mC8krb
/mNau66rXn4hDbb4 /Downloads/Verus-CLI-Linux-v0.7.2-6-amd64/Verus-CLI-Linux-
v0.7.2-6-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AV6kEwABQR85s2Rs3BjLBikfd0gDQ3CeUxJwRDzx0Y+DocPn5SzWdxq6W45PVDVDFBRPUzKY48mC8krb
/mNau66rXn4hDbb4
f20c5e9620b483721396485e8005b8ac64ed832d8a05c6b56665f0d5a3b584ca
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY CLI upgrade v0.7.2-5, with fixes that only apply to testnet. Mainnet features are unchanged from v0.7.2-4.
This release includes the following improvements and updates:
- Fix testnet transaction funding issue for multi-currency `sendcurrency` commands
- Fix testnet currency conversions

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a

protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/9b115c8bbad22397b6f211ab41c4b0cfb7dd99fd675e5680f6b0e6b4154eaefc/detection
Linux-AMD64:
https://www.virustotal.com/#/file/530b4b20799f1f4a4ed814c0a899740ae97fce39bb42067b3fb983c

ca37a9f3c/detection
Linux-ARM64:
https://www.virustotal.com/#/file/e774acb721f294f2ac355514994d98564a57691e23b415811f4e9d2cd9d88ef3/detection
Windows:
https://www.virustotal.com/#/file/74e812b08df6e9c1499ea3eef06267c31a502fedb09584b4693875c013a617cf/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ATh2EwABQSCZ/hlR7AeMdT6iPkL2OZXZVk5dQ66E8CXWq05376CHdCDoXi26dqXxiOQrLTaX+jDZ3uBU
C3nTBGm0MN5pFKSM /Downloads/Verus-CLI-Linux-v0.7.2-5-amd64/Verus-CLI-Linux-
v0.7.2-5-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ATh2EwABQSCZ/hlR7AeMdT6iPkL2OZXZVk5dQ66E8CXWq05376CHdCDoXi26dqXxiOQrLTaX+jDZ3uBU
C3nTBGm0MN5pFKSM
ebf3f8b2209c9a2e76f6f18f0aeba70532d3766e6a49d2cf5a165e1ea1ba02bb
```

The result will be true if the signature is valid.

```
true
```

v0.7.2-4 WITH THE WORLD'S FIRST FRIENDLY-NAME, zk-SNARK ADDRESSES FOR MAINNET AND TESTNET

Announcing NON-MANDATORY CLI upgrade v0.7.2-4 with the world's first friendly-name private z-address support for sending and receiving funds and messages, including the following improvements and updates:
- `sendcurrency` can now use sapling addresses as native currency funding sources and destinations for any transaction.
- An identity's z-address can now be referenced as a funding source or destination by appending `:private` to an ID (eg. `"idname@:private"`). This form of z-address will also be able to be used with the z_getbalance command.
- New wildcards can also be used as funds sources in the 'sendcurrency' command and parameters to z_getbalance that include:
- All transparent addresses in the wallet (both R-addresses and IDs)
`"*"`` - All R-addresses in the wallet"R"- *All IDs controlled by the wallet"i"*-sendcurrency`will now be an asynchronous API that will return an operation ID, as`z_sendmany`does, and complete in the background. It will also have its status available through`z_getoperationstatus`.

NOTE: Due to the zero knowledge nature of z-addresses, a z-address is an endpoint in your VerusID which can be changed if you revoke and recover an ID, funds held on a private address or

endpoint are not able to be redirected to the newly recovered ID unless you still have control over the z-address itself. Transparent funds controlled by an ID are recovered when the ID is revoked and recovered, even if you have lost the keys. This is a fundamental limitation of zero-knowledge privacy, not an error and not something to be fixed. Please consider this when using friendly-name private addresses as funds and not just message endpoints.

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API,

`getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
```

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

```
```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset Instructions

Please delete the following directories before launching the testnet:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c7dfc60b6ec87eb96f53d4961123be6a13a08f94a862005fee8ea27a1b67f7b1/detection
Linux-AMD64:
https://www.virustotal.com/#/file/34c5155bb6dc2229340d8a9c38a2272a10961e89453d89e9199c367fc15d2267/detection
Linux-ARM64:
https://www.virustotal.com/#/file/22b6ac8cbbf8f0fc99344ab397df14b031bdcc9ede3975ca644e1b8ace2d9496/detection
Windows:
https://www.virustotal.com/#/file/5ea98824ef1ec2d6cd144fcc01c86fc56cf9652fcdbd9f7f956fbe2850da93cf/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AUdCEwABQR/riPCH6AtcsHHlq1S+hzLqdIbwswLsMNZlQvZM3zuXVBBP2LHlILu26ry+0/
XRNtdX152Ym2lANInbT13470eK /Downloads/Verus-CLI-Linux-v0.7.2-4-amd64/Verus-CLI-
Linux-v0.7.2-4-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AUdCEwABQR/riPCH6AtcsHHlq1S+hzLqdIbwswLsMNZlQvZM3zuXVBBP2LHlILu26ry+0/
XRNtdX152Ym2lANInbT13470eK
ece5000684a135c841397d6a382ef1f643dcf52f4f8fabce2e66b9d86007af6f
```

The result will be true if the signature is valid.

```
true
```

Announcing NON-MANDATORY upgrade v0.7.2-3 with the following improvements and fixes:
- Fixes a rare issue in wallet synchronization for IDs that have been transferred between wallets in a specific combination of transfers and spends of prior outputs. In the cases seen, the wallet maintained some outputs as unspent, when in fact they were spent on the blockchain. We have seen a very small number of people hit this issue after transferring IDs between owners and/or separate wallets and having spent prior outputs on transactions that still have valid UTXOs to the ID. If you have encountered such an issue, this version will properly synchronize the wallet with a full rescan. If you have not, use of this version will prevent the issue from ever occurring.
- Stricter updateidentity paramater checks to ensure that revocation and recovery IDs are already registered, valid IDs on the blockchain before accepting a change

Testnet Reset

The testnet was deleted and relaunched on v0.7.2-2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2-2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset Instructions

Please delete the following directories before launching the testnet:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/84c176f9730b6105c7e8c75e91dc2750bc836a99d1ff7424199217b3b97c2285/detection
Linux-AMD64:
https://www.virustotal.com/#/file/15c8e29508364f247f6475a0e73472fedbb6c726f563fa0c0fe9d176674eed79/detection
Linux-ARM64:
https://www.virustotal.com/#/file/17c2388076edcd86a631badb4efc6bc27994fc62472b08ed82026377b7dfce3b/detection
Windows:

https://www.virustotal.com/#/file/e15b4dfe2bf1e8099e58ba5557cf89342b3e412bb111a035d6a06ab957e257a2/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AZ3ZEgABQSBWDFw2fO8xPGO+g25zaO/DZRoS8p4zs/1lvfiEVMqi9j1pP/UbkGNfjzB4292fVkT4BSA/
4uh9pxw6ngQJG/w1 /Downloads/Verus-CLI-Linux-v0.7.2-3-amd64/Verus-CLI-Linux-
v0.7.2-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AZ3ZEgABQSBWDFw2fO8xPGO+g25zaO/DZRoS8p4zs/1lvfiEVMqi9j1pP/UbkGNfjzB4292fVkT4BSA/
4uh9pxw6ngQJG/w1
82d1a30f848bb5703080c70027ad1475f97129b17da96f8ee89de7dde2f53b4a
```

The result will be true if the signature is valid.

```
true
```

Verus v0.7.2-2 is a mandatory for pools and notaries, but not generally mandatory Verus node and wallet upgrade, with support for the new Verus testnet (VRSCTEST) with fully decentralized finance, advanced ID capabilities, and DeFi liquidity system integrated into the new network protocol. While most of the changes in this release are focused on testnet, the mainnet support does include some improvements and a fix for the getblocktemplate API used by pools

Testnet Reset

The testnet was deleted and relaunched on v0.7.2-2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2-2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of

stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee

distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```

```

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

```
```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.

Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset Instructions

Please delete the following directories before launching the testnet:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/84c176f9730b6105c7e8c75e91dc2750bc836a99d1ff7424199217b3b97c2285/detection
Linux-AMD64:
https://www.virustotal.com/#/file/746bcadd639964ff0782f81f38fce1c3bd3d6c5be36660ca9710e44e69d99a1a/detection
Linux-ARM64:
https://www.virustotal.com/#/file/035a8df1be988dbed30c8f1a95b88836d6a4454cf4779269409e70751cb5c67b/detection
Windows:
https://www.virustotal.com/#/file/64ea435716eb01e5ac278bd70b0d47a583b20b063848429c7a436bbfb63cd184/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AV1AEgABQSAsdDnDMsHH8QbwB3LZGCNFuQofET4ZKnEAJsiLbm74kTJyG7Vr3qk5DwMQF3ThcoaYbWKf
G1npYRtus5cijbqc /Downloads/Verus-CLI-Linux-v0.7.2-2-amd64/Verus-CLI-Linux-
v0.7.2-2-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AV1AEgABQSAsdDnDMsHH8QbwB3LZGCNFuQofET4ZKnEAJsiLbm74kTJyG7Vr3qk5DwMQF3ThcoaYbWKf
G1npYRtus5cijbqc
9d4a31ebce0ea8e720a590d16af04d5790da12ec08e6a13858cb0048bab97a73
```

The result will be true if the signature is valid.

```
true
```

This update is required to continue participating on the testnet launched in v0.7.2

Verus v0.7.2-1 is a non-mandatory but highly recommended Verus node and wallet upgrade, with fully decentralized finance and liquidity system activated on the new Verus testnet. While most of the new capabilities in this release are focused on testnet, the mainnet support does include some performance improvements and also fixes an issue that may occur in an edge case of revocation and recovery of IDs, which can result in a revoked ID being irrecoverable from an earlier version than v0.7.2.

Testnet Reset

The testnet was deleted and relaunched on v0.7.2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or

more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
```

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

```
```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset Instructions

Please delete the following directories before launching the testnet:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8de6ae82139b99f0fbea6d48e3dbe1659b134bb9cb48d053eec1b79b040efb07/detection
Linux-AMD64:
https://www.virustotal.com/#/file/719f95e355b4429cffaf43e573e0bb517042790cb79f6c926e58fc57c7cda9f6/detection
Linux-ARM64:
https://www.virustotal.com/#/file/00e095873cf6975dd9fd4ad3fa53b0960c745f65edd45ae7b9b0dcd8fe91499a/detection
Windows:
https://www.virustotal.com/#/file/4a95a6ecbfff65f758c0c64af2712d679e99e5dd5d52a50a321d5a7cceb645c6/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AWUvEgABQSCHGrmIokvjGdV5CIVcarS7VKfkYyBOm1iCJWkqb/gZS06XyEwk3cQa9wuRdqH0Fs4rpwOw
6QLFmxVwoIWOD6Qe /Downloads/Verus-CLI-Linux-v0.7.2-1-amd64/Verus-CLI-Linux-
v0.7.2-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AWUvEgABQSCHGrmIokvjGdV5CIVcarS7VKfkYyBOm1iCJWkqb/gZS06XyEwk3cQa9wuRdqH0Fs4rpwOw
6QLFmxVwoIWOD6Qe
3d2b27e47ffac9a5eb5053853cf4b7175a289fb850f6a040dcb92cdffe0a388f
```

The result will be true if the signature is valid.

```
true
```

We are excited to announce Verus v0.7.2, non-mandatory but highly recommended Verus node and wallet upgrade, with fully decentralized finance and liquidity system activated on the new Verus testnet. While most of the new capabilities in this release are focused on testnet, the mainnet support does include some performance improvements and also fixes an issue that may occur in an edge case of revocation and recovery of IDs, which can result in a revoked ID being irrecoverable from an earlier version than v0.7.2.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a

process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

```
verus -chain=VRSCTEST getcurrencyconverters btc eth
```

Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

```

verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'

```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"VRSC-BTC"}]'
```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via","VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
  verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"alice@","amount":10, "convertto":"NEWCOIN",
"preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":97,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"minpreconversion":
[10000,5.1298]}, "initialsupply":[20000]'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset Instructions

Please delete the following directories before launching the testnet:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/9bbb4067fc3b047fd95acd1fa3504a3e6fd6af6aa78510c251b73caf0f7d7557/detection
Linux-AMD64:
https://www.virustotal.com/#/file/335e72dd777655cec07bf470a315b607f15cf7447e356a41fd997fcb1854cf90/detection
Linux-ARM64:
https://www.virustotal.com/#/file/7f81d7ec26702be237a02de8c6fab63c776e36683ff10ce5638ce92bf5f3a40b/detection
Windows:

https://www.virustotal.com/#/file/78318a873f9ffb4d620df9768fc369ca1dea1603288b5259769b4372ed979cdd/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ARgrEgABQSBGafKV5NB9DxKiq3p45BQwC/+qkka54hIUejd6c4hP1BjSiUi1lris2Z3LACS7HyjOjLIN
KwXL4qvWnPSwLTmG /Downloads/Verus-CLI-Linux-v0.7.2-amd64/Verus-CLI-Linux-v0.7.2-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ARgrEgABQSBGafKV5NB9DxKiq3p45BQwC/+qkka54hIUejd6c4hP1BjSiUi1lris2Z3LACS7HyjOjLIN
KwXL4qvWnPSwLTmG
e7cee9dc4786b02cc9756a63f783a876d7a3fedd7127ca6d1cc54f1418a919a0
```

The result will be true if the signature is valid.

```
true
```

New in v0.7.1-5

This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or later. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-5 has no improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet

for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-5 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will

typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.

Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8114d40ea20883a728640905f282e4017da8b3f98d373ab8230be68ba6f3a89a/detection
Linux-AMD64:
https://www.virustotal.com/#/file/8344a259a176124f44255a01344fa40917ee217b2d260f6f14480ebec94dd75f/detection
Linux-ARM64:
https://www.virustotal.com/#/file/592a3983cc35519f52001309db326e7a6e114a754aa4dfd75dbb738eb512879a/detection
Windows:
https://www.virustotal.com/#/file/8c52846d8e26223a95214ff01cde849ab5a7c94105aafe4cb8e66c537c8c35b8/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AXAkEQABQR/tMlFb0UpI3f6Y610spPi50zVSGpWwzNV49QF0sUavujInLrw3Ej9wt6iPf/
KrjpSLELJVcH8UWXBRkm4Yhh6b /Downloads/Verus-CLI-Linux-v0.7.1-5-amd64/Verus-CLI-
Linux-v0.7.1-5-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AXAkEQABQR/tMlFb0UpI3f6Y610spPi50zVSGpWwzNV49QF0sUavujInLrw3Ej9wt6iPf/
KrjpSLELJVcH8UWXBRkm4Yhh6b
e70862ae89f0d7968fee0d19e730e74917a3f322da8e1e9f45a2484bc9a95aa7
```

The result will be true if the signature is valid.

```
true
```

New in v0.7.1-4

This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or later. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-4 has no improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-4 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must

have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/baba5b1731867dd6af01894a33cdb82e22a3c5970f7e55a1a277e3ee1378d7ba/detection
Linux-AMD64:
https://www.virustotal.com/#/file/0b2dbe6de69f18d631330d51e78a22e5e59df43ae49f05bad0eaabc4b91581f8/detection
Linux-ARM64:
https://www.virustotal.com/#/file/f34ee571cfd090d151bd46965d6585eaa339386af43df23ae0b20c040517c0f3/detection
Windows:
https://www.virustotal.com/#/file/a27703796965270bbe1c8067b5a33f445375ebc08b78b7c33ca2f41ef09f075f/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AVEGEQABQSDgax8VEtroSX/nc2mIh9s7iipkyP6zfItclRbbsGyX9zctzA7pluFyEb1LDpJ70fReiaZ7
hu6OxcdK6HsSU7Nv /Downloads/Verus-CLI-Linux-v0.7.1-4-amd64/Verus-CLI-Linux-
v0.7.1-4-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AVEGEQABQSDgax8VEtroSX/nc2mIh9s7iipkyP6zfItclRbbsGyX9zctzA7pluFyEb1LDpJ70fReiaZ7
hu6OxcdK6HsSU7Nv
d3d04a488ab18ebfd758b06e9f8462e1ff657de6a78258752efc4c5f9d46463f
```

The result will be true if the signature is valid.

```
true
```

New in v0.7.1-3

- This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1 and 0.7.1-2, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or "-2". You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.

- Improved staking supply calculation

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-3 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next,

"convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-3 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.

So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one

named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/66a86cd778033c507263d787b5577f43ad99e8362f6d44437480265ef1e9c062/detection
Linux-AMD64:
https://www.virustotal.com/#/file/c654c929ad3ee93fda0bdbbd6f42d8f296231bccd119b0bae62b874e82c54bb5/detection
Linux-ARM64:
https://www.virustotal.com/#/file/ac2a202cda3e6531f078d644a4f81d5e31b09368f1f21cc69ff06be0a31ebba2/detection
Windows:
https://www.virustotal.com/#/file/516776ea4c48dffec7f140c0f328587d292612187685fe78b007a60269de0600/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AcYAEQABQR8mgNgm19xS1rpDzEZnHPKUiTW5uzw4/mTbLae/Ir/dbDKtpCF+ACXWBww7NGIKZc6y87V4
Jz2jAgU9A2l8sgQf /Downloads/Verus-CLI-Linux-v0.7.1-3-amd64/Verus-CLI-Linux-
v0.7.1-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AcYAEQABQR8mgNgm19xS1rpDzEZnHPKUiTW5uzw4/mTbLae/Ir/dbDKtpCF+ACXWBww7NGIKZc6y87V4
Jz2jAgU9A2l8sgQf
3987a67ee8cb2b7f41ac53b2d6137979fb9bd4d35a246b6549192ba55f77d716
```

The result will be true if the signature is valid.

```
true
```

New in v0.7.1-2

- This non-mandatory release is compatible with mainnet with no changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-2 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.

So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one

named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/1f9cb1fe4ba5aef4a0859aebd2b89001c72e1482fd27a99e1ee67f8de11011a7/detection
Linux-AMD64:
https://www.virustotal.com/#/file/ccc03d8a038bf21af83d873f3f814024b54f4cdbf3113f37f19ed8e7b2885a84/detection
Linux-ARM64:
https://www.virustotal.com/#/file/857f8e01a0ef59f779ec7c19f97a4399067399e1b28a616a3ad51a93eca0b707/detection
Windows:
https://www.virustotal.com/#/file/bbc7fddee170a738fa066e56547c5804e444fe51962fdac565a14e5a4d6cc199/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQm0EAABQR8S/gEslZAXyvOmsbZ1iHKTDXFlABnJNPMCUXXNzwd57xW3IHjcumslSyi+JKkkhbWX6eXx
kFcMICW8wQRE7i8k /Downloads/Verus-CLI-Linux-v0.7.1-1-amd64/Verus-CLI-Linux-
v0.7.1-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ARrVEAABQSC8iC2EZWSLJ1VbjWVUkiQtKAcZCuq0GTiGd2BK4WIJ/k1H1a/3gSoTTD+L38mt9TaIHi2E
KcH2CBLTQ6fmtdGb
9c8e0dcc69d9cc604a0c1fd43c82d6eb965aaf52aed2c8d770f8241e25be021b
```

The result will be true if the signature is valid.

```
true
```

New in v0.7.1-1

- Re-launched VRSCTEST with fixes. Make sure you delete your VRSCTEST data before connecting to the testnet.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-1 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/e94319a3ad2b81825b0ce231deb1117ac28ae897ecee1dfb0fe424c198db7d06/detection
Linux-AMD64:
https://www.virustotal.com/#/file/a9eb0f6a0bec1feb8e156410e0ccdbb7139e16a4b10bdea5e939b30ad834f50f/detection
Linux-ARM64:
https://www.virustotal.com/#/file/f17a5af4d09123d3f11f3682929fdc92f46f474463134fbdc1c2da4e56bdf8f6/detection
Windows:

https://www.virustotal.com/#/file/aafa51f1d6068d65a80715db668e7e7e39b9d9437005de957d08f7300094a20c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQm0EAABQR8S/gEslZAXyvOmsbZ1iHKTDXFlABnJNPMCUXXNzwd57xW3IHjcumslSyi+JKkkhbWX6eXx
kFcMICW8wQRE7i8k /Downloads/Verus-CLI-Linux-v0.7.1-1-amd64/Verus-CLI-Linux-
v0.7.1-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ATbAEAABQR/kpmoY0p915MY2NhXkl7O/wiDt/cjVfBKuvd0CdV5a4AFX5CnctBuOMEwCyiowyzYNPDan
zSzE277ZDbMNH5We
95e70c52caeafffb0c9715aa28588068c9178c89fca2f60313074db052022a33
```

The result will be true if the signature is valid.

```
true
```

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are

simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.
Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must

have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/7a46a8f579b45a08eaf9a20f138ff75c80590740959803c8c93a5f54810af3ad/detection
Linux-AMD64:
https://www.virustotal.com/#/file/26cb5bd36f7dcd4c8db49baaf427afea0aa87dc518dfac3bac93950d204944f9/detection
Linux-ARM64:
https://www.virustotal.com/#/file/2b07c7505f0c67d8957420677f85bb2a1947029494b01c2707fdb3d0c2644aaa/detection
Windows:
https://www.virustotal.com/#/file/1bf02b7b5022f3b4e9b072e604a6a94656a9dc9188d1625e557fe304bdf04067/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AQm0EAABQR8S/gEslZAXyvOmsbZ1iHKTDXFlABnJNPMCUXXNzwd57xW3IHjcumslSyi+JKkkhbWX6eXx
kFcMICW8wQRE7i8k /Downloads/Verus-CLI-Linux-v0.7.1-amd64/Verus-CLI-Linux-v0.7.1-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AQm0EAABQR8S/gEslZAXyvOmsbZ1iHKTDXFlABnJNPMCUXXNzwd57xW3IHjcumslSyi+JKkkhbWX6eXx
kFcMICW8wQRE7i8k
26cb5bd36f7dcd4c8db49baaf427afea0aa87dc518dfac3bac93950d204944f9
```

The result will be true if the signature is valid.

```
true
```

At block 1053660, the Verus Mainnet Protocol upgraded to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash that is now live on Verus mainnet.

The network upgrade activated the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new “sendcurrency” API for the command line provides more control when sending from and to multiple addresses or identities.
- “updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.

- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.

- Staking and hashing consensus protocol updates with VerusHash 2.2

New in v0.7.0-4

- Validate ID destination on `sendfrom`, `sendtoaddress`, `sendmany`, and `z_sendmany`

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/97fec5a44e617452d48e13f0c00736430d904a72d85fc61731fd01452de77880/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3c8134361c965ab87751f817a097c1825ca61e2a521bfd1ed9382fbc44638920/detection
Linux-ARM64:
https://www.virustotal.com/#/file/c5c70239f6cb94cd1ab8cb109597f609a198f1bce644fa3434f50426f9fa8724/detection
Windows:
https://www.virustotal.com/#/file/418d4da1a4113b29f7c7ce490f8aa527d6217f2d85fd10ff639b33110ab5eba0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AR1XEAABQR8XbEix6eohYN9m6BbLjcdUkCZH/NYw1/mfBq/mIT6SewPg4PvaJHsUIO8FLB8kQDpa9eKg
lWJN9DTfvpWlMH9Z /Downloads/Verus-CLI-Linux-v0.7.0-4-amd64/Verus-CLI-Linux-
v0.7.0-4-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AR1XEAABQR8XbEix6eohYN9m6BbLjcdUkCZH/NYw1/mfBq/mIT6SewPg4PvaJHsUIO8FLB8kQDpa9eKg
lWJN9DTfvpWlMH9Z
3c8134361c965ab87751f817a097c1825ca61e2a521bfd1ed9382fbc44638920
```

The result will be true if the signature is valid.

```
true
```

At block 1053660, the Verus Mainnet Protocol upgraded to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash that is now live on Verus mainnet.

Although the network is running smoothly after the upgrade, an issue was identified and resolved in this release. In addition, a couple of user features, including restoring "-pubkey=hexpubkey" as a parameter for redirecting staking and mining rewards are included.

The network upgrade activated the following features:

- Added support for season 4 Komodo notary node operators

- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.

- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.

- A new `sendcurrency` API for the command line provides more control when sending from and to multiple addresses or identities.

- `updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.

- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.

- Staking and hashing consensus protocol updates with VerusHash 2.2

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/08254d35e8693eb304b35d1cee55f03761b78ad76dc5e25d6e42b300fcdd4e50/detection
Linux-AMD64:
https://www.virustotal.com/#/file/aa4842153c54510626b171128265059da30ae2bbfca974ceecc4348abb67e544/detection
Linux-ARM64:
https://www.virustotal.com/#/file/8a23406c8726ccf2fced09a20f298590d334ea00dbbca02eeadfae3eb8349d95/detection
Windows:
https://www.virustotal.com/#/file/d274514424e2334a7695e74c20569e9267158f22303021539060da6d82ab482c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AfQaEAABQR9cFt4qNU1KD14hlvClLhfICrm9LWuzcyW8Wh+zsu/BOQji9pUp0PRBnEEH4aQWlfRgeS4j
2sByVDix3VMwj2aM /Downloads/Verus-CLI-Linux-v0.7.0-3-amd64/Verus-CLI-Linux-
v0.7.0-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AfQaEAABQR9cFt4qNU1KD14hlvClLhfICrm9LWuzcyW8Wh+zsu/BOQji9pUp0PRBnEEH4aQWlfRgeS4j
2sByVDix3VMwj2aM
d509506edcf48a3ba64d814916f98e46105a0f6b437635cf5d2f2247e1574d7a
```

The result will be true if the signature is valid.

```
true
```

MAKE SURE TO UPGRADE BY June 14TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE.

At block 1053660, the Verus Mainnet Protocol will upgrade to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash will go live on the Verus mainnet as well.

The network upgrade will activate the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new `sendcurrency` API for the command line provides more control when sending from and to multiple addresses or identities.

- `updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.

- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.

- Staking and hashing consensus protocol updates with VerusHash 2.2

Notable Changes:

- Fix Komodo S4 notary count

- `sendcurrency` does not require coinbase funds to be shielded.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/74ccab3336fc62efb8fb4cf1a37f3c50eba59030c906bf5475227e9276d29e4b/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3cc1b3fd86368bb6a6c69f385e3411495974f4920dda96d9e8e6d48ef3aa0143/detection
Linux-ARM64:
https://www.virustotal.com/#/file/300379f4530e3a7f9379a55ffb3bb88ff7c1cb7806421bbfa8f3c1bb588bece7/detection
Windows:
https://www.virustotal.com/#/file/56dfa302af9a6185406c3db7102b75c77e67187d8d674b3c47374f8c28b90056/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AfcIEAABQSBXNoiaxzhPSEo99oIkVNEWvQP1ebMA5l4TNOlsUzi+DjoO9AcsQjAG9gF/
HuVJlS5gAgaQN28Pwj6ExYWoY/dm /Downloads/Verus-CLI-Linux-v0.7.0-1-amd64/Verus-
CLI-Linux-v0.7.0-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AfcIEAABQSBXNoiaxzhPSEo99oIkVNEWvQP1ebMA5l4TNOlsUzi+DjoO9AcsQjAG9gF/
HuVJlS5gAgaQN28Pwj6ExYWoY/dm
0587c3da3d0f64015be98e097353a24d0581b9fbc1f3a788adf31a408c643023
```

The result will be true if the signature is valid.

```
true
```

MAKE SURE TO UPGRADE BY June 14TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE.

At block 1053660, the Verus Mainnet Protocol will upgrade to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash will go live on the Verus mainnet as well.

The network upgrade will activate the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new `sendcurrency` API for the command line provides more control when sending from and to multiple addresses or identities.
- `updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.
- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.

- Staking and hashing consensus protocol updates with VerusHash 2.2

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/67f38974426e598b60f3cc97ca4a0fd80f8291e24c6738ecea59ddf0055172e1/detection
Linux-AMD64:
https://www.virustotal.com/#/file/29919bf5117ec7aa88b86eae07e83d203909cd92a8cbb0a02911633e761cdff0/detection
Linux-ARM64:
https://www.virustotal.com/#/file/a9d5f4cff70ba5b7be4d236ae692affa846336a44f6ba1a398742d87fe3ec352/detection

Windows:
https://www.virustotal.com/#/file/7a7634d3bcc29b9a10190fd58a37be53ef686f7cb80293a0c4feb6aed9a056b4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
Afn5DwABQSBdcb9HEXsztsaPQLtxhwA1U31x7ahYamF+DqxIn3j+pk/WQEv79ewYd5pDgjp5AJ/
cpKWIfCVi468j1PbZ5WeJ /Downloads/Verus-CLI-Linux-v0.7.0-amd64/Verus-CLI-Linux-
v0.7.0-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
Afn5DwABQSBdcb9HEXsztsaPQLtxhwA1U31x7ahYamF+DqxIn3j+pk/WQEv79ewYd5pDgjp5AJ/
cpKWIfCVi468j1PbZ5WeJ
f769349015e0f4b8a6da9b071e3342f2043bd6eec35a69a97de7c7b4fd910148
```

The result will be true if the signature is valid.

```
true
```

This is a mainnet release with fix to prevent any recurrence of the recent minor fork at block 1002417 and integrated support for the new PBaaS testnet. This release is approved for use on mainnet nodes, and it is being released on Github prior to full GUI wallet release v0.6.5-2 to enable exchanges, pools, and infrastructure nodes to update immediately, as needed.

Notable Changes:

- Added `getsaplingtree` API
- Added support for Verus Testnet with PBaaS Multi-currency Tokens
- Resolved multi-block reorg issue on mainnet that can prevent successful reorg in rare, otherwise valid cases

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Verus PBaaS Multi-currency Tokens (TestNet only)

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a

business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/9221506334ae6562d31f441202a5cbc019e6790ebec06c61ddaedebd9425db92/detection
Linux-AMD64:
https://www.virustotal.com/#/file/2e03e9442d34fe118cbeb586a5cff0567e98cdb4c6bda7cedf110f7456f7c62c/detection
Linux-ARM64:
https://www.virustotal.com/#/file/ac5d54de262512ff02405da2d6182fe4741d3929d8218a333024e2649e42123c/detection

Windows:
https://www.virustotal.com/#/file/f5223182d7dc042bf0cfc383586d34327d3cd3c0955fa0db92812d41146f2298/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted text file.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AfeNDwABQR+hVp1D3jZymlkW4NrwdJbsh4jQDxxXf44WNf6QZoXY/UiU1WFy9RM+/pkCC1IQfCXF6I9e
y3B/3DofrvzKKAsV /Downloads/Verus-CLI-Linux-v0.6.5-2-amd64/Verus-CLI-Linux-
v0.6.5-2-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AfeNDwABQR+hVp1D3jZymlkW4NrwdJbsh4jQDxxXf44WNf6QZoXY/UiU1WFy9RM+/pkCC1IQfCXF6I9e
y3B/3DofrvzKKAsV
99d7451472bc1fb34359d17e524d7755defbd279b0c2569a67231f8b7a37f9ff
```

The result will be true if the signature is valid.

```
true
```

This release is intended for use in the VRSCTEST only. Mainnet approved testnet release is not yet available

Notable Changes:

- Check for creation of invalid staking transactions

Verus Testnet PBaaS Multi-currency Tokens

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option.
To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/64f4f431c73cc8d28da48ee5adaa8fddb07ca20339706c1aa953abe8819cbbd4/detection
Linux-AMD64:
https://www.virustotal.com/#/file/bfa5cdbbf35495f9e6e4c8a71da7a05c41795c844aecf0d05817f7e5ee32f2aa/detection
Linux-ARM64:
https://www.virustotal.com/#/file/3c71f41cf03ea85fbc3d7c46b6fae18b9458bdd66d06f9dd16eea3a4dc0b3e05/detection
Windows:
https://www.virustotal.com/#/file/c671dff603e3a0219c9bd8403b8ac1efddb0485621dfce9b802e975b4d7652cc
/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Testing@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted textfile.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Testing@"
AcBbDwABQSAhpO83zww8L/K3dZTImWLcF8D/u93GyeQFCtKRoiZbb10cvNCMVDN/13S74mZUGCekQQV9
t4fzChTy1mJ/DKKk /Downloads/Verus-CLI-Linux-v0.6.5-amd64/Verus-CLI-Linux-v0.6.5-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Testing@"
AcBbDwABQSAhpO83zww8L/K3dZTImWLcF8D/u93GyeQFCtKRoiZbb10cvNCMVDN/13S74mZUGCekQQV9
t4fzChTy1mJ/DKKk
b589a80b79a022f24524e113ccd5b7841d137fb198c5e64618c7a3ed867feb62
```

The result will be true if the signature is valid.

```
true
```

This release is intended for use in the VRSCTEST only. Mainnet approved testnet release is not yet available

Verus Testnet PBaaS Multi-currency Tokens

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus

protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/f0cfd8bcd39b69e23481d265fb8789d27ae6e655ee168acb06573268ded74185/detection
Linux-AMD64:
https://www.virustotal.com/#/file/eb259e23a629cac31789621cf4e917905ac43ba722e47f82f8dc0360d4f6ba72/detection
Linux-ARM64:
https://www.virustotal.com/#/file/ea0b8244bb8c6ddea4eb0eac51e50aa9dfd57ebb0c17696f44c3e21b3d99320c/detection
Windows:
https://www.virustotal.com/#/file/80bc1d1c574a91708cbb4f13be9ba110cccc0c4fd8555bc4378737660eef6d06
/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted textfile.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AT9MDwABQR+giVSMbRiq3p60hCUVfEdm/yPLZYjX1Eh+ufvtnpDPhyjvjUGQRClss+mQohhXPxfWY5oq
IzfioqJsnHHlo2XH /Downloads/Verus-CLI-Linux-v0.6.5-amd64/Verus-CLI-Linux-v0.6.5-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AT9MDwABQR+giVSMbRiq3p60hCUVfEdm/yPLZYjX1Eh+ufvtnpDPhyjvjUGQRClss+mQohhXPxfWY5oq
IzfioqJsnHHlo2XH
60020bfebbb2ced560c5d1e1333c43d9f80c4db4797812213ae04ad39df99c22
```

The result will be true if the signature is valid.

```
true
```

New in v0.6.4-3

- Added `z_viewtransaction` api: Gets detailed shielded information about an in-wallet transaction

Verus CLI v0.6.4-3

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs

are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN

ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c93d7f3ee35502a2dd9eaf87d76b253c6ee8e1b62c256103a267082daf3fa7f9/detection
Linux-AMD64:
https://www.virustotal.com/#/file/4346c377e8a6d95aae7abb289703a32ee1288786586a01fd0da6595c34d33ac9/detection
Linux-ARM64:
https://www.virustotal.com/#/file/131ec86c7a4024648d33c80992cfbd1708743fd4d1b9087234df6410d34ecb91/detection
Windows:
https://www.virustotal.com/#/file/6d57ec0805dc0f56604be718c83e61fe57fea9e640fcc3fb28e2bc8784619864/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".
1) Extract downloaded archive
2) Verify signature for the extracted archive using the extracted textfile.
3) Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ATPPDgABQR/xBtY9wU+RmBJqX5Y6AWdPlw9nJShSAFpyGzu50ufFR3pTC1s1Wyk96zGTCZqMUzaurrbt
u4ysrKZNr7FUq/O/ /Downloads/Verus-CLI-Linux-v0.6.4-3-amd64/Verus-CLI-Linux-
v0.6.4-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ATPPDgABQR/xBtY9wU+RmBJqX5Y6AWdPlw9nJShSAFpyGzu50ufFR3pTC1s1Wyk96zGTCZqMUzaurrbt
u4ysrKZNr7FUq/O/
3e5cc7bea46c6869545a0a886d66fdbf21fb2800761e3dca6f626046681ee26e
```

The result will be true if the signature is valid.

```
true
```

New in v0.6.4-1

- Added `convertpassphrase` api, allowing wif key exports of electrum wallet passphrases

Verus CLI v0.6.4-1

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/21dff79a0060325787c96b4aaaf9becd161fe958cb4428adbc82d9bd1edb07cc/detection
Linux-AMD64:
https://www.virustotal.com/#/file/c4d339cc8a85b3beb240b3ea4692a7630124441305cf07cf7890724277df67f7/detection
Linux-ARM64:
https://www.virustotal.com/#/file/f8e92c5acf1fe5e0f317d4d0bbc8288c03b2ce4c21791a33ff5f2de60c385ca8/detection
Windows:
https://www.virustotal.com/#/file/d4c883ff9aa87e4e0356e66d44952a50e7bbff6fd8120a93b799510e20644a5a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AcFjDQABQSDvcPZH+EBJHnQR491hRjINjwGtaVf3S8wFwNAY0KwYF1Qj3z0kndcLIJKqDCVKWj5RarHj
CApwUGbjYK4x29Uc /Downloads/Verus-CLI-Linux-v0.6.4-amd64/Verus-CLI-Linux-v0.6.4-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AcFjDQABQSDvcPZH+EBJHnQR491hRjINjwGtaVf3S8wFwNAY0KwYF1Qj3z0kndcLIJKqDCVKWj5RarHj
CApwUGbjYK4x29Uc
0cbd162064918fb63a8914c0cf963e0eb310fcd100daa7433a9066c6806491e1
```

The result will be true if the signature is valid.

```
true
```

New in v0.6.4

- Estimated active staking supply displayed in output from `getmininginfo`
- Average block fees for last 100 blocks displayed in `getmininginfo` output
- Immature coins, including time locked coins displayed in `coinsupply` output
- Eligible staking balance, with consideration for all disqualifying factors displayed in `getwalletinfo` output
- `z_exportwallet` as wallet backup now accepts boolean parameter to remove all addresses with no UTXOs or Identities in the backup
- Staking performance is improved
- Fixed rare condition of improper ID wallet synchronization of transferred or revoked IDs

Verus CLI v0.6.4

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c90815f194641646e069824711fca2dc1f7a9cd6ba5bd5016bdcdcbff8a1f190/detection
Linux-AMD64:
https://www.virustotal.com/#/file/5ea61dee317688aee255f6f5255d62d0307517196098ace354881a49ee36a3c3/detection
Linux-ARM64:
https://www.virustotal.com/#/file/3e9e463afed0eaa737a6fbd51438d5e828fccd4ca3d4be74751e79fe3eecc119/detection
Windows:
https://www.virustotal.com/#/file/a3b8980fa5044177976e53e30bb3142c9ed78d982fede9d4cf015417dfa35f8a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AYpdDQABQR9QSLDca3cCXdpZhNtUyVL3GdFM4248ka0Nhpp0d+/yAGmH7pmBXfcyObnInU5cxGjgbe5n
2wLZRKmllS/zPM6q /Downloads/Verus-CLI-Linux-v0.6.4-amd64/Verus-CLI-Linux-v0.6.4-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AYpdDQABQR9QSLDca3cCXdpZhNtUyVL3GdFM4248ka0Nhpp0d+/yAGmH7pmBXfcyObnInU5cxGjgbe5n
2wLZRKmllS/zPM6q
60df4018e329cc6748379369d02943b1ff4e7b21363beedb2624ea2770fba1b9
```

The result will be true if the signature is valid.

```
true
```

Notable Changes

- Improved coinsupply API speed and added accurate immature funds count

Verus CLI v0.6.2-1

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash

algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/4cf0e9684029faaa29b87277cbe19f85b40a059d6a04ae9861d94bebb6d95c56a/detection
Linux-AMD64:
https://www.virustotal.com/#/file/6bb4ebb2c4d5e0b59a52fb060d4b06596843955b3814dbf1c858a3

e5d6be2ea9/detection
Windows:
https://www.virustotal.com/#/file/bd2fd47bb22dbe81bb34d74c04182d03e7e72f83a0cb254acb0b869d8ee07091/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AZPmDAABQR/rW5EvSr1wxFw4FP+FBgTbNoZJKROpreBeH8dGQx4YJFcjz40LDSBqWfufFFEFmaZGWNNW
Y3QVIp3jwU7qyLfu /Downloads/Verus-CLI-Linux-v0.6.2-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AZPmDAABQR/rW5EvSr1wxFw4FP+FBgTbNoZJKROpreBeH8dGQx4YJFcjz40LDSBqWfufFFEFmaZGWNNW
```

```
Y3QVIp3jwU7qyLfu
6bb4ebb2c4d5e0b59a52fb060d4b06596843955b3814dbf1c858a3e5d6be2ea9
```

The result will be true if the signature is valid.

```
true
```

Verus CLI v0.6.2

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash

algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/32742e2b62aa446f6772ae68706626a953652c1766ac118cfb612ac6707bc48d/detection
Linux-AMD64:
https://www.virustotal.com/#/file/0eb57c65fc05daa34a55863f12987d4127cd8dd8c8fdce36df206e9

00598c9fe/detection
Windows:
https://www.virustotal.com/#/file/9aab61dc575c25e71df5c1c6d8c1eeb56d0b818e6a5d18378a412a7e929c810a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AfLZDAABQR9H2eDMg7vKCrf7KmJt/4+8Vi4FMJSrer93IUNCMhkw63JHnCXrIv1GysI3nhkY3qoCXGpW
QiIBlk6pzA/+Ztg8 /Downloads/Verus-CLI-Linux-v0.6.2-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AfLZDAABQR9H2eDMg7vKCrf7KmJt/4+8Vi4FMJSrer93IUNCMhkw63JHnCXrIv1GysI3nhkY3qoCXGpW
```

```
QiIBlk6pzA/+Ztg8
0eb57c65fc05daa34a55863f12987d4127cd8dd8c8fdce36df206e900598c9fe
```

The result will be true if the signature is valid.

```
true
```

Notable Changes

- z_importviewingkey and z_exportviewingkey for Sapling to enable confidential, verifiable, and transparent voting

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next

year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/e1884e04a3d5932b99e3bfcb8a34cd3ed4034340367a5fade0b831eda6a0daa2/detection
Linux-AMD64:
https://www.virustotal.com/#/file/b514dbcf8ae804efe16ea5080751e3c0e6702a6db59b38eb7994c9a5fdc81c4c/detection
Windows:
https://www.virustotal.com/#/file/b5fb64cad0bb29ab76b0215b7050beee2919bcb1b6eef6bd52436d665fa3a051/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

Notable Changes

- Fix accretion of coinbases in mempool

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and

associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/6dc2eb44ee294829c6f14e38349ccd1f8ba2bf683b93c0cda328c9bbdd588bab/detection
Linux-AMD64:
https://www.virustotal.com/#/file/ba3014f6ae1298fea11258474eb053b8841b1be04db911e4a3e2f80990ac42c3/detection
Windows:
https://www.virustotal.com/#/file/596ef34389d9aa801936da73f39f7ac6c85878f9b927321537e850deae4f614f/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

Notable Changes

- Improved staking check

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/387f4933627c7ad774c19ede52aba9d2ca65b27dca49e6dc6ee5bc471cbebcb0/detection
Linux-AMD64:
https://www.virustotal.com/#/file/fc157e8826f9f547e8ff54eed6ccd1cf8c314104e90a46f39464a9484c9597fb/detection
Windows:
https://www.virustotal.com/#/file/ff10726405536584fa9be5c2989c8365aaa5eaa79bb5bd6cd603d9e2f979e8dc/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

This release fixes an issue that does not generally affect the Verus network, but can corrupt individual wallets in the case of a specific type of reorg occurrence that is relatively rare, but can happen. We STRONGLY RECOMMEND that all users upgrade to version 0.6.0-6 asap.

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it

made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c4415da8270a48944cd29507129085a677ea09ba3cf014f93393a9f6fe08391c/detection
Linux-AMD64:
https://www.virustotal.com/#/file/599b02ca94c9409f4167c15f7c5203b909f38c43693df702bb0dd92d7cdbe098/detection
Windows:
https://www.virustotal.com/#/file/75b8aee40a5394f6b99ab0ac815b8973cc475d83bedb6e24cb7e37996553bd7c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
THIS UPDATE FIXES AN OVERLY AGGRESSIVE BLOCK REJECTION IN THE VerusID PROTOCOL, AND IS A MANDATORY UPDATE FOR THE UPGRADED VERUS MAINNET

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WAS UPDATED TO INCLUDE VerusID. THIS UPDATE IS THE LATEST UPDATE FOR THE UPGRADED VERUS MAINNET

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and

associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8d9a74da37c9b1be5f416c2f2f92f772b086da01770d8ed6a118778f87a7e568/detection
Linux-AMD64:
https://www.virustotal.com/#/file/27aa9d8d4a89dadf1a87928664eef7dc70ef92624d62596ee6e699b064f17daa/detection
Windows:
https://www.virustotal.com/#/file/e8f9b060d37e7028714d8c078705c8d6ceaef51b0d6f4c121cfbe40ab781b16f/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Alleviate peer banning behavior

- Staking enhancements

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WAS UPDATED TO INCLUDE VerusID. THIS UPDATE IS THE LATEST UPDATE FOR THE UPGRADED VERUS MAINNET

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs

are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a3c9e986b441a4973c15a3578fb713578b294b19077feedb2182a78270560d52/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3075c0ebc2337e36ce4d815a6a0d2a2715dcea2118cf7b70822061c60264c7e3/detection
Windows:
https://www.virustotal.com/#/file/594cf99ecc390e580edfbec4f7c73f45c959515030d256afb16e07074ba142e4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- This update fixes an issue which would ban nodes too aggressively, resulting in disconnection from the chain and potential to fork away from the main chain. This should be considered a mandatory update.

- Add cansign and canspend to getidentity

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

**AT BLOCK 800200, THE VERUS MAINNET PROTOCOL HARDFORKED. THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET.

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a82b8cfb4207b6a8e05300bc91c0363dbad0de7e3f5a1a57043b641ba4852b2e/detection
Linux-AMD64:
https://www.virustotal.com/#/file/1246d439115b7578ddd47f8427a93f3ebad9a637d0735ed1db31003a9a3f53be/detection
Windows:
https://www.virustotal.com/#/file/2ca4b8bef8ad3e23bb896593be1509cf85619f2a6b87e72f9e59b99b52b63810/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Enable optional VRSC or VRSCTEST chains as parent of an ID. For example, bob@ is equivalent to `bob.vrsc@` or, on testnet, `bob.vrsctest@`

- Update stratum protocol for mining to include solution information for miner, which enables smooth autoswitch and prepares for PBaaS merge mining

- Minor issues, including listidentities issue on Mac and compatibility with pool mining, even on block numbers under 16 to prepare for PBaaS

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and

associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8a7f7bf3df09fa352979b1b92b3eea99fc802bce41f4349cfdeb3f25792233ce/detection
Linux-AMD64:
https://www.virustotal.com/#/file/de2961c3d4c0a00ce565989721d25b3abcf1e9e70b511c5cdb4e343005b13cc4/detection
Windows:
https://www.virustotal.com/#/file/74bbe68954498821f079a9a611cf6b8b8f8deb10f9a6e2feb594deba5c5ed7b1/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
Thanks to @hellcatz and his constant development, improvement, and support of LuckPool.net for Verus and other coins they mine, we have fixed an issue that would have allowed the old hash algorithm to be used after the VerusID and VerusHash 2.1 network activation.

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while

still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/021e59635bb77ccb8de45b017f5e07b1d71ab55e8f508e3a68aca96f7c9c2f5b/detection
Linux-AMD64:

https://www.virustotal.com/#/file/2c39e4fc74ba9a1874d5d3b9d957543c83ecca50f80f0be8f844b6a599a8b65c/detection
Windows:
https://www.virustotal.com/#/file/9f6a11ab3621496a0210d531167351a66c7b7881dfb405872b82db73c9094efe/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm.

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated

keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/bfa8df164e7ac5a1e7ea10e7abec275059df247fd9d62258e32b28777c40baf4/detection
Linux-AMD64:
https://www.virustotal.com/#/file/644b4e30c681f2268f18e9e258c696a8ab4ae53964bca4b0a2d256e407b464ec/detection
Windows:
https://www.virustotal.com/#/file/f9ad88116f33a1cc27ff75a4eb6108873a4ab3820759d0998b7a3a0e69d4d9b3/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Set content hashes will be a key/value map instead of an array

- Improve invalid block and invalid transaction handling.

Verus ID and VerusHash 2.1 Network Upgrade

This is the third release candidate (RC3) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.
This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today's centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate. This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into

secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary -** this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.

    - **Revocation -** this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.

    - **Recovery -** this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.

3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.

4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@”`.

5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID

- 20 VRSC to the referrer’s referrer’s ID

- 20 VRSC to the referrer’s referrer’s referrer’s ID

- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
```

```
That will respond with something like:
```

```
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
```

```
    "name": "Name",

    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",

    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",

    "parent": "",

    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"

  }

}
```


You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {

    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/512b039b4cfd1a561372c0858c1c592efcab9a7d0bcba9fc2a9f2a298e77bb8c/detection
Linux-AMD64:
https://www.virustotal.com/#/file/d3ab563b621a6953be5b0e3405d27608425f8fd7491d7dd0fcb347d0b535d265/detection
Windows:
https://www.virustotal.com/#/file/8f6b826b008ad66397cd4091a52fccc0ff1c8b3b54e2f7f73249e1ef92296f73/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Improved Identity synchronization
- Cleaned up debug log output
- Set block space limit for Identities

Verus ID and VerusHash 2.1 Network Upgrade

This is the second release candidate (RC2) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this

release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.
This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today's centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate. This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which

will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and

unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.

- **Revocation -** this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.

- **Recovery -** this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.

3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.

4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@”`.

5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID

- 20 VRSC to the referrer’s referrer’s ID

- 20 VRSC to the referrer’s referrer’s referrer’s ID

- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
```

```
That will respond with something like:
```

```
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
```

```
    "name": "Name",

    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",

    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",

    "parent": "",

    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"

  }

}
```


You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {

    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/fac7a41048e010a5f7bc4b61c121fec0b73316a9c35929cfc0beae353b4ca843/detection
Linux-AMD64:
https://www.virustotal.com/#/file/8aa4b6435a25338c150316c5bbb2c6cb6650f3e48e2d9bfe4c0c95a9a2784c53/detection
Windows:
https://www.virustotal.com/#/file/186b0a524d2ece4f3a6360050a3cb9c883ca239cdf6b82e99046b45f0c5a6e75/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verus ID and VerusHash 2.1 Network Upgrade

This is the first release candidate (RC1) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.
This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today’s centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate. This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.

- **Revocation -** this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.

- **Recovery -** this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.

3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.

4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@”`.

5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID
- 20 VRSC to the referrer’s referrer’s ID
- 20 VRSC to the referrer’s referrer’s referrer’s ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
```

```
That will respond with something like:
```


```
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
```

```
    "parent": "",


    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"


  }


}
```


You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {


    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/7f1762616b2a7221c256f02ebebd6724f7b490887f86835e7093701e93720c00/detection
Linux-AMD64:
https://www.virustotal.com/#/file/a350815413d6279b1b09c332185536a5d7f8225c588d27cb99cb354aeee659e4/detection
Windows:
https://www.virustotal.com/#/file/e58fd29884468b97d8320bbfe960b0fee3439e9025f0e78fc358dfb45465f4aa/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
This release is intended for use in the VRSCTEST network. Mainnet use is still being tested.

Notable Changes

- Add blockchain rule to consider any identities without primary addresses or nonzero minimum number of signatures as invalid.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary -** this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.

    - **Revocation -** this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.

    - **Recovery -** this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.

3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.

4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using "z_getbalance identity@".

5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100

VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer's ID
- 20 VRSC to the referrer's referrer's ID
- 20 VRSC to the referrer's referrer's referrer's ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
```

```
That will respond with something like:
```

```
{

  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",

  "namereservation": {

    "name": "Name",

    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",

    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",

    "parent": "",

    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"

  }

}
```


You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/5cc801f027d86a59d4d70de91697fa86d96ca5034f957c81c0a2f7895a5a5677/detection
Linux-AMD64:
https://www.virustotal.com/#/file/0a989ab93e059fc778475498459b9de0484557ee8f44feb2fe418c58d976c38a/detection
Windows:
https://www.virustotal.com/#/file/2a661915722798ba65ca84990707c4f6cea1f390a79f717adc83c8b50861b581/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
This release is intended for use in the VRSCTEST network. Mainnet use is still being tested.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.

    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.

    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.

3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and

recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.

4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using “z_getbalance identity@”.

5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID
- 20 VRSC to the referrer’s referrer’s ID
- 20 VRSC to the referrer’s referrer’s referrer’s ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
```

That will respond with something like:


```
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction,

including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/23ae5a14f89a0806f1a6ac143c01a2a46bc3ddbe39136534444824e02b94329c/detection
Linux-AMD64:
https://www.virustotal.com/#/file/82bf9054feaf39e750a98542086bf5e5217ce18200bd8ff40e509bbb4db00ac3/detection
Windows:
https://www.virustotal.com/#/file/e18d521cf928025dc935a6b280909cd4aaf06ed77fb60a0af5f43f92ddeb2805/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

IMPORTANT SECURITY/PRIVACY UPDATE

Version 0.5.9 incorporated a fix for the Zcash protocol issue described in: https://z.cash/support/security/announcements/security-announcement-2019-09-24/

Updating to v0.5.9 or more recent version is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Fixed address parsing in getrawtransaction

Testnet Reset

The testnet was deleted and relaunched on the 0.5.9 release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO 0.5.9, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol
.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
```

```
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
```

```
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/13dbc2ac7d64f5bf507187b343359c6afc082138bd37d27ae42180b1e5150e49/detection
Linux:
https://www.virustotal.com/#/file/d6e6b789457ba789bc98385bc444d16c2d8d56f1fe04beeb6fe2bdfadb36b664/detection
Windows:
https://www.virustotal.com/#/file/c0490311446822c8667a202cad64d90539a45ed84da8a90d7d616df4bbcfe995/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

IMPORTANT SECURITY/PRIVACY UPDATE

Version 0.5.9 incorporated a fix for the Zcash protocol issue described in:
https://z.cash/support/security/announcements/security-announcement-2019-09-24/

Updating to v0.5.9 or more recent version is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Avoid re-indexing whenever possible

Testnet Reset

The testnet was deleted and relaunched on the 0.5.9 release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO 0.5.9, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
```

```
"%APPDATA%"\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol
.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/341d9d06eed1c8df0057d56f853667e9738e8ec89cd6329a84c81543d38582fe/detection
Linux:
https://www.virustotal.com/#/file/eaf521b19ff4c6d8bfc84d25a59ae6ac745003f3927ed3fc71d5a6af883b32c7/detection
Windows:
https://www.virustotal.com/#/file/10e6bb5be7b7aa33bb57beb6105a5477e8f863c8f3d2b487f239ede9864dbbfc/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

IMPORTANT SECURITY/PRIVACY UPDATE

This update incorporates a fix for the Zcash protocol issue described in:
https://z.cash/support/security/announcements/security-announcement-2019-09-24/

In addition, this version also includes updates to Sapling functionality, including the Sprout to Sapling migration APIs added to the Zcash protocol up to and including version 2.0.7-3 of Zcash. The issue fixed by this update does not expose any funds to theft or counterfeit, but it does fix a vulnerability that an attacker may exploit to probe for information regarding a specific node's control over private keys. Since the issue resolved does not put funds directly at risk, this update is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Integrated Zcash updates from Sapling release to 2.0.7-3
- Added Sprout to Sapling upgrade API
- Including support for PBaaS multi-chain test network with permissionless blockchain launches, reserve currency conversion protocols, merge mining, and easy cross-chain transactions.

Testnet Reset

The testnet will be deleted and relaunched for this release. **IF YOU HAVE PREVIOUSLY LAUNCHED VRSCTEST, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
```

```
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
```

```
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
%APPDATA%\Komodo\VRSCTEST
```

```
%APPDATA%\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol
.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
```

```
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
```

```
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/341d9d06eed1c8df0057d56f853667e9738e8ec89cd6329a84c81543d38582fe/detection
Linux:
https://www.virustotal.com/#/file/eaf521b19ff4c6d8bfc84d25a59ae6ac745003f3927ed3fc71d5a6af883b32c7/detection
Windows:
https://www.virustotal.com/#/file/10e6bb5be7b7aa33bb57beb6105a5477e8f863c8f3d2b487f239ede9864dbbfc/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Properly distribute initial contributions on non-reserve chains
- Show chain name on failure to notarize
- Continue attempting to notarize after running out of notary rewards
- Fix getdefinedchains currency state

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol
.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
This release includes the ability to connect to and use the Verus Testnet with easy multi-chain network, cross-chain send, and in-wallet currency conversion capabilities that go beyond any blockchain system or cryptocurrency available today. Don't take our word for it. Just give it a try when you're not using the other mainstream wallet features

Try out cross-chain sends, create and exchange your own reserve currencies and merge mine up to 15 chains at once on the VerusCoin testnet!

Notable Changes

- Added Testnet support for Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol
.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/462a2014bdd120eb155b964fe1f22910671ade38e26ff4372e64341e9613dff5/detection
Linux:
https://www.virustotal.com/#/file/6b1b3b848ab12c471d0469db37e5f8059aee565550d9e1c38fd97a55d493f184/detection
Windows:
https://www.virustotal.com/#/file/878ee7b0ed79d20f03994cf6322345be4a9f857aa4326ceb564d4ec51edd53ef/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact

price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Notable Changes

- Minor fix to export transactions

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Notable Changes

- Fixed daemon output export error message on VRSCTEST

- Fixed fees bouncing back from a reserve send to Verus and allow exports to continue if done incorrectly

- Include currency state in chain definitions for both getchaindefinition and getdefinedchains

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Notable Changes

- Added command `refundfailedlaunch` to refund all contributions made to PBaaS chain to VRSCTEST after a failed chain start block.

- Fixed wallet scan issue that could prevent a wallet from recognizing reserve coins sent to it on a PBaaS chain. If you have such a wallet, a rescan with the new update should make your coins appear.

Chain definition:

- Maximum of 3 eras

- Minimum 500 blocks worth of notarizations.

- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
```

```
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
```

```
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Notable Changes

- Removed redundant debug output

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/f469ceee35e3abd616c3323907af5019379b45618f35693113f55cbf42afdaaa/detection
Linux:
https://www.virustotal.com/#/file/5d2a68970ec9e8eeaabde49f719e95342506ee46e37589094fed96e369a3f84e/detection
Windows:
https://www.virustotal.com/#/file/bd26b50c5ce9a085e02fe6ec76f8d2c16d2b190d16493606f9690485aad22eac/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Notable Changes

- Removed redundant debug output

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/4de5aa22de6d43554f3456abe690c4ca1405e3326748dda14710d85f66634919/detection
Linux:
https://www.virustotal.com/#/file/4f5bc9cc22046b675fea304155026cc2c951ad4760ee5a2e27b42b5db8e4765d/detection
Windows:
https://www.virustotal.com/#/file/787d46b30681e86c20d1875ab3c18c3253168a680c801c64de7a9f43b8b2db2c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
This release does not support the PBaaS testnet. To participate in the Pbaas testnet, please use the PBaaS Cross-chain Technology Preview.

Notable Changes

Renamed komodod and komodo-cli to verusd and verus and removed wrappers.
Minor staking improvements.
Fix longest chain reporting
Fix getchaintips bug
Preparations for Pbaas

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/3a04de7202aff21afb8005d462b9d5acf577bae8705ec308de56c720f40841d5/detection
Linux:
https://www.virustotal.com/#/file/3351d8fd49ac074605d776f147c0906d149ac936418512c1940465f233d8f9c0/detection
Windows:
https://www.virustotal.com/#/file/d1bdee20f375dad722b71851f69588062cd9e3760ed653957096080de9d0b3d0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). Although this version does also work with Verus mainnet, it does not support PBaaS on mainnet and is not an official release. As always, use at your own risk. Do not use as main wallet

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras": [{"reward" :
10000000000}], "notarizationreward" : 1440000000, "billingperiod" : 1440,
"nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd-chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). Although this version does also work with Verus mainnet, it does not support PBaaS on mainnet and is not an official release. As always, use at your own risk. Do not use as main wallet

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras": [{"reward" :
10000000000}], "notarizationreward" : 1440000000, "billingperiod" : 1440,
"nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd-chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). Although this version does also work with Verus mainnet, it does not support PBaaS on mainnet and is not an official release. As always, use at your own risk. Do not use as main wallet

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras": [{"reward" :
10000000000}], "notarizationreward" : 1440000000, "billingperiod" : 1440,
"nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd-chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). Although this version does also work with Verus mainnet, it does not support PBaaS on mainnet and is not an official release. As always, use at your own risk. Do not use as main wallet

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras": [{"reward" :
10000000000}], "notarizationreward" : 1440000000, "billingperiod" : 1440,
"nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd-chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Approximate release height was set too low on the last release. Please be sure to upgrade your VerusCoin daemon and/or Agama wallet to avoid disconnecting from the network

Linux Dependencies
Ubuntu 16.04/Xenial:

```
sudo apt install libcurl4-openssl-dev
```

Ubuntu 18.04/Bionic+

```
sudo apt install libcurl4
```

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a358156eaf210f594c214a309d6c201d0ccf9e7806065563b483a22e80dff791/detection
Linux:
https://www.virustotal.com/#/file/e745633c99499a3cf968bc2f2c2d2fc56ecdc8d2996a2cf4fb43feff452acb71/detection

Windows:
https://www.virustotal.com/#/file/2e3cbb2b9f38bccbf49061dce480d267c35f7d7ae5172fcda2c3a5afefd86736/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
Please note that this is a test release intended to use with test chains. We do not advise using it for main net.
Do not use as main wallet

Sample chain definition:

```
./komodo-cli -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras":
[{"reward" : 10000000000}], "notarizationreward" : 1440000000, "billingperiod" :
1440, "nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are needed in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./komodod -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Please note that this is a test release intended to use with test chains. We do not advise using it for main net.
Do not use as main wallet

Sample chain definition:

```
./komodo-cli -chain=VRSCTEST definechain '{ "name":"SOMECOIN", "eras":
[{"reward" : 10000000000}], "notarizationreward" : 1440000000, "billingperiod" :
1440, "nodes" : [{"nodeaddress" : "127.0.0.1:2500", "paymentaddress" : ""},
{"nodeaddress" : "127.0.0.1:2501", "paymentaddress" :
"RDyDzC5JB66Y29fzBkamywqEn1PJXTzj8B"}]}'
```

Maximum of 3 eras
Minimum 500 blocks worth of notarizations.
Minimum of 0.01 VRSCTEST per block of notarization.
VRSCTEST are not paid out yet, but they are needed them in wallet to make a chain.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./komodod -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

IMPORTANT: CRITICAL UPDATE -- UPGRADE IMMEDIATELY

This release fixes a security vulnerability discovered in a dependent library

MAKE SURE TO UPGRADE TO 0.5.6 OR GREATER ASAP.

Linux Dependencies
Ubuntu 16.04/Xenial:

```
sudo apt install libcurl4-openssl-dev
```

Ubuntu 18.04/Bionic+

```
sudo apt install libcurl4
```

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/d873c1b7ccb6dc1a7df28d301c28cb810bc9df6c328f90b3165933b9985a0307/detection
Linux:
https://www.virustotal.com/#/file/e5074eda5ad79e17ae766d67896d3d0e218aef6db4054b5f442be6a679f49209/detection
Windows:
https://www.virustotal.com/#/file/ebf07aacc896b4acb3aa39655f4ad370af1ccb3220199db9f01f5251d6867949/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
v0.5.5c introduces a portability fix for a fault on some CPUs that did not support optimized mining when attempting to mine. This allows mining on CPUs that don't support optimizations, although they will be relatively slow.

MAKE SURE TO UPGRADE TO 0.5.5 OR GREATER ASAP. See 0.5.5 release for details

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/bb689d033f8cd9c6c5a81d0f5aa473cbd47956e56faff3ba31488f76c9d1d638/detection
Linux:
https://www.virustotal.com/#/file/9e42f4313be2b03eaaef4d8add50766663dc53caff7953d02ddffff92cc344b4/detection
Windows:
https://www.virustotal.com/#/file/370b69673e6f6226dcf4ecd262aa62c859b7d2f2d8466afe5f4f6fe8cc0f7313/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/
v0.5.5b introduces bug fixes for staking status not reverting to true after performing a z-transaction..

MAKE SURE TO UPGRADE TO 0.5.5 OR GREATER ASAP. See 0.5.5 release for details

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/d96bd2afb5db178dbc07cbe1708892e3d08c6b7984c32a0064a3dcfb2d2943dc/detection
Linux:
https://www.virustotal.com/#/file/1b08617478b2b84c3d1181a6f93f9bc6359ac57e2c434acd8e57d8b603988a9f/detection
Windows:
https://www.virustotal.com/#/file/040fa02e57c2e55fd0f5fc70ef8b8ea7c38e54c10fa16e967a147c9614d1fe50/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

IMPORTANT: CRITICAL UPDATE -- UPGRADE IMMEDIATELY

COMBINATION SECURITY FIX AND INITIAL PBaaS PREPARATION

During an audit, a potential security issue was recently identified and fixed. This version both fixes the security issue in question immediately, and also uses this as an opportunity to include some early PBaaS functionality that while not fully enabled, will allow us to test future PBaaS functionality on the main Verus chain in the near future. There is no activation date for these fixes. They are important and immediate. There has been no exploitation of this issue yet, and to our knowledge, details are not known outside of core developers. We recommend immediate upgrade of all Verus wallets.

Increased number of outbound connections from 4 to 5.
Fixed CLI bug displaying 0 mega hashes done when it was hashing just fine.
Fixed zfunds calculation on coinsupply RPC
Fixed bug preventing Mac builds from mining with the enhancements introduced in v0.5.3

Linux and Windows only release, MacOS on the way, Mac users should remain on version 5.1
To upgrade your wallet, run the agama-upgrade script included and enter the directory of your Verus Enhanced Agama app
Fixed one not fully portable call in the portable code

IMPORTANT

THIS IS A MANDATORY UPDATE, AND IF YOU DO NOT UPDATE, YOU WILL BE UNABLE TO ACCESS THE MAIN VERUS CHAIN AFTER BLOCK 310000.

VerusHash 2.0

New **VerusHash 2.0**, designed to close the performance gap between CPU, GPU, and FPGA.

Verus Agama Improvements

Full support of any Komodo asset running natively alongside Verus Coin. This is done by separating out the daemon for Komodo, allowing both the Verus daemon and the Komodo daemon to run simultaneously.

https://medium.com/veruscoin/veruscoin-0-5-0-official-release-notes-6d371095dc7a
v0.4.0h fixes address indexes for the insight explorer and some APIs, as well as a rare segfault in network reorg edge cases from the cheatcatcher.
Version 0.4.0g includes all improvements in 0.4.0f and an expiry on stake transactions to allow deletion from wallets later
Version 0.4.0f includes all improvements in 0.4.0e.
In addition, it provides more detail about stake (winner of the contest) and mint (staked reward) transactions in Agama GUI as well as CLI listtransactions. It also introduces safeguards against any possibility of double staking behavior caused by forks and reorgs and the ability to set the maximum number of transparent and private transactions in Agama in settings->App config.
0.4.0e fixes issues and introduces a much cleaner view of staking transactions and minted coinbases in the Agama GUI. The following community reported issues are resolved:
1. Coinbase shielding problems to Sapling and sometimes to Sprout private addresses are resolved
2. Staking will again default to sending the coinbase to the address that staked the reward unless **-pubkey=66charpubkey** is set. The public key of an address where you would like the coinbases to go can be determined using **"validateaddress"** either in the CLI or GUI.
3. Agama GUI no longer shows ins and outs or multiple lines for a staking transaction. Instead, it now displays an easy to understand single line for each "stake" transaction and also easily distinguishes between mined and minted coinbases once they are mature.

0.4.0c - released 10/29/2018

- All versions of the Verus Coin wallet from **0.4.0** and later have support for **Sapling high performance zero knowledge transactions**, which will activate on the Verus Coin Network at block **227520**. In addition to basic Sapling support, 0.4.0c includes support for Sapling compatible, transparent mobile transactions. **For lite mode usage on the Verus Coin or Zcash networks, you will need Verus Enhanced Agama v 0.4.0c.**

- In addition to Sapling and support for Stake Guard transactions, which all versions **0.4.0** or later have, 0.4.0c also includes a powerful *"Cheat Catcher"* that can earn money as miners and stakers who use it protect the Verus Coin network by catching stakers who are staking for rewards in a selfish way that does not contribute to converging the network on one blockchain. **The new "Cheat Catcher" support requires a Sapling z-address, either passed in to the CLI wallet with the "-cheatcatcher=zaddress" option, or added in the Settings->App config of the enhanced Agama GUI.**

- Once your cheat catcher address has been set, a **full node wallet that is mining OR staking, will also be looking for forks where the same winning coins may be posted on two different forks, signed by the same person.** If this is ever discovered, the the mining or staking wallet will create a transaction that has cryptographic evidence of the cheating and submit it for shielding to the cheat catcher's address. That means that while staking or mining, anyone who has a Sapling cheat catcher address set, may periodically find an additional shielded coinbase or two in their cheat catching address. **Earning cheat catching rewards does not new create coins, but actually takes them from cheaters, so DON'T CHEAT :) !** Earning a cheat catching reward can also occur whether or not you earn a block reward and is available equally across the network to all miners and stakers, more dependent on who happens to observe the cheating fork and catches the cheater than on staking or mining power.

Version 0.4.0c also fixes the following issues reported in prior versions:

- Wallet balance can get out of sync with blockchain when staking

- Some non-Latin usernames cause failure to load

- When used to load KMD in GUI Agama, Komodo stops accruing interest after transfer

IF YOUR WALLET IS OUT OF SYNC, YOU CAN RESCAN TO FIX THE PROBLEM.

For CLI:

- Simply add the parameter **“-rescan”** when starting the daemon.

For GUI:

- Close the application, wait for the GUI to close, then restart the GUI

- When restarting, select **“Activate Coin”**, not a dropdown shortcut
  on the “Activate Coin” screen, select **“Verus”** and select **“Native mode”**

- Choose the **“daemon param: rescan”** option
  click **“Activate Coin”**

Fixes:

- Solves an issue where pool may exceed getblocktemplate's ability to keep up with work queue when network is under stress

- Fix to verus bash script to allow execution from external directories

Fixed:

- Komodo compatibility

Verus Coin 0.4.0 - Release Notes

IMPORTANT

IF TWO OF YOUR MACHINES STAKE BLOCKS WITH THE SAME COINS, EACH ON A SEPARATE FORK, YOU CAN NOW LOSE YOUR BLOCK REWARD TO A CHEAT-CATCHER.

STAKING IS NOW OFF BY DEFAULT TO PREVENT MULTI-MACHINE STAKING OF THE SAME COINS OR ADDRESSES. YOU CAN EASILY ENABLE STAKING WITH THE “-mint” PARAMETER TO verusd, “setgenerate true 0” WITH “verus” CLI, OR THE STAKING BUTTON IN THE GUI.

ONLY STAKE ANY ADDRESS OR WALLET ON ONE COMPUTER AT A TIME, NOT MORE, UNLESS YOU CAN BE ABSOLUTELY SURE THEY ARE IN SYNC AND ON THE SAME CHAIN AT EVERY BLOCK. YOU CAN STILL MINE ON AS MANY MACHINES AS YOU’D LIKE.

Sapling

Verus Coin 0.4.0 includes support for Sapling addresses and transactions, including memory usage reduction of near 98% and performance improvements over previous z-transactions typically 40-

50x. As with Zcash Sapling upgrade rules, older “Sprout” addresses and “Sapling” addresses may not be mixed in the same transaction. That means if you have Verus Coin that you would like to move from an older Sprout z-address to a Sapling z-address, you must first send it to a transparent address, then to the Sapling z-address. Sapling addresses can be used for shielding coinbase transactions or other functions. Retrieving viewing keys for Sapling addresses, which is not required for normal transactions, is not yet supported, and will be included in the next Sapling technology update.

Crypto-conditions and StakeGuard

Verus-enhanced, Komodo crypto-condition technology, enabling smart transactions and new types of on-chain functionalities. Verus has enhanced the original crypto-condition technology to make integration with standard transactions easier and enable things like StakeGuard, the Verus nothing-at-stake solution. In later releases, the same technology will enable PBaaS, voting, identity capabilities, and more.

Cheat Catcher

In addition to Stake Guard, which supports the nothing at stake solution, this version of Verus includes an early, functional version of the Cheat Catcher, a system that enables every miner or staker to catch stake cheating, when someone stakes two blocks with the same coins on two or more forks. With this version, all known orphaned stake transactions are compared to successful stakes when they mature, and if the orphan was trying to stake the exact same coins but on a different fork and the real fork at the same time, the miner of the block that is 100 blocks later, when that coinbase matures, will mine or stake a spend transaction for that cheat into the block, and send it to a sapling address the miner or staker has set in their wallet. If your wallet catches and spends a cheat transaction the reward for that coinbase will be automatically shielded to your address at the same time as you win the block. The early Cheat Catcher does not broadcast cheating transactions, and it will not catch all possible cheats on the network. It will make it a statistically losing proposition to cheat. This means you should not stake the same addresses on more than one computer at a time. Doing so creates the possibility that you will stake on two forks with the same coins, which puts your earnings at risk. Future cheat catcher implementations will excel at catching cheats on the network.

Stake Delegation

In the new Verus network protocol that activates at the same time as Sapling, you can now set the parameter -pubkey in the command line wallet, which will redirect both mining AND staking rewards to the public key of an address, which can be obtained from an address with the “validatekey” command.

- Timing improvements for more efficient staking
- Improved support for synchronizing with non-VRSC KMD-family coins
- Non-mandatory

This release should make a huge difference in global network synchronization, mining, and staking, which should provide a strong foundation as the network grows. In addition, it has significant efficiency improvements for both mining and staking. Here is a rundown of the improvements in this version:
New in both command line and GUI:
1. Reconfigured connection ratios, better seed node discovery built in. Default outgoing connections now 4, default incoming for full node 384. If your machine can handle it, you can help the network and broadcast your stakes more quickly to increase the number of connections in your VRSC.conf with "maxconnections=n", where n is the number you wish to use.
2. Improved lag detection. If your wallet detects a lag, it immediately stops attempting to mine or stake and devotes maximum resources to catching up. Rather than continuing to mine or stake, which may contribute to further lag, this action ensures mining and staking occurs on a viable tip of the chain.
3. Improved staking efficiency. This version raises the default priority of the staking thread, ensuring that the staking process is fast and efficient, even in cases where it would have also lagged with mining or other machine load.
4. All updates and improvements from Komodo Platform as of 1.5 weeks ago have been merged. Preparing for unlocked era.
First release with verus-cli
Linux, Mac and Windows variants built and distributed for the VerusCoin Launch.

© 2020 The Verus Foundation. Website generated with MDwiki © Timo Dörr and contributors.

VerusID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. **VerusIDs** provide plenty of opportunity for identity applications. Specifically, **VerusID** provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- Revocable -- each ID includes a revocation authority, which defaults to the identity self. If another ID is specified as the revocation authority it can be used to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft or turnover in an organization.

- Recoverable -- each ID also includes a separate recovery authority, which also defaults to self. If another ID is specified as the recovery authority it can be used to recover the ID from its revoked state, with the option to alter the primary authorities used to spend and sign.

- Private - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to

provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong identity, attested to with a quantum secure signature when that is available.

- Powerful - Multiple addresses or other IDs can be defined as primary addresses, and any number of those may be required to spend, sign, or alter the identity (N of M). The revocation authority may only be altered by the revocation authority, and the same applies to the recovery authority, either of which may be another identity with its own N of M multisig controls for its primary addresses.

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked

completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

How-To¶

Guide to install Bootstrap for your Verus Wallet.

- Important General Information
- Verus-Desktop Procedure
- Verus CLI Procedure
- Optional:
- Advanced Issue Resolving Procedure
- Optional:

Attention: Read it completely before using.

Important General Information

Verus `Wallet.dat`, Chaindata & `VRSC.conf` standard locations

- Linux: `~/.komodo/VRSC`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`
- Windows 10: `%AppData%\Komodo\VRSC\`
- OS independent through Verus Desktop: Click `help`, `Show Verus data folder (default)`

Tip: The easiest way is to copy the location above and paste it into your address bar of your file browser. Your operation system will accept the input, interpret where that location is and bring you there.

Verus-Desktop Procedure

1. In Verus-Desktop, exit your profile, by selecting the exit icon at ther top right.
2. Wait a minute or two, allowing the wallet to close completely in the background.
3. click `help`, `Bootstrap VRSC`. That opens up a new window.
4. Follow the instructions and when finished successfully, select your preferred profile and enter Verus Desktop.

Verus CLI Procedure

1. Go to the folder where your daemon is extracted (standard `verus-cli`)
2. Shutdown `verusd` and wait for it to close completely
3. doubleclick `fetch-bootstrap` in your file browser.
4. Follow the instructions and when finished, start your `verusd` daemon as usual

Optional:

Watch this video with an explanation how to accomplish the steps above: [Bootstrapping your wallet using the supplied script]

Advanced Issue Resolving Procedure

Necessary files:

Link 1: [Download latest Wallet]
Link 2: [Download Verus Bootstrap]

Procedure:

1. Make sure your wallet is not active.
2. If you already had you wallet running, backup essential files:
   a. Go to `VRSC Wallet location`
   b. copy `wallet.dat` to a *SAFE* location
   c. copy `VRSC.conf` to a *SAFE* location
   d. Verify that both files are copied to your safe location
3. Make sure the latest version of your Wallet for Verus is installed
   a. Download the latest Verus Wallet from link 1, supplied above.
   b. Verify the SHA256 checksum & signature of your download, to verify you have an untampered installer.
   c. extract the file you just downloaded to a suitable location.
   On MacOS and Linux you will have extracted an **AppImage** which can be run directly.
   Windows users need to run the **installer**.
4. Installing the bootstrap:
   a. Download the bootstrap from Link 2. (For Windows a zip archive is available to accomodate native extraction. Linux and MacOS users can use the tar.gz archive)
   b. (Optional, but recommended) Verify the md5, sha256 or sha512 checksum and the signature of your download, to verify that you downloaded an untampered Bootstrap

archive.
c. Remove all files and folders from `VRSC Wallet Location` except `wallet.dat`, `debug.log`, `VRSC.conf` and if applicable `VRSC-bootstrap.*`.
d. Extract the downloaded archive to `VRSC Wallet location`. Make absolutely sure the folders `blocks` and `chainstate` are extracted into the correct folder. If the end up in a different folder (eg. `VRSC-bootstrap`-folder) move them to `VRSC Wallet location`.

5. If you had a VRSC wallet running before, restore essential files:
   a. Go to `VRSC Wallet location`
   b. Verify that your `wallet.dat` is bigger than the one in this folder (if any is present)
   c. copy `wallet.dat` from your *SAFE* location
6. Start your wallet

If you followed these steps, you will have installed/updated the latest version of a wallet for verus, made a backup of your wallet and installed the bootstrap. If desired you can remove the downloaded bootstrap archive to free up space on your hard drive.

Optional:

Watch this video with an explanation how to accomplish the steps above: Bootstrapping your wallet manually

Information compiled by Oink.vrsc@.

Guide to verify a new wallet download.

Important General Information

Wallet download page: https://verus.io/wallet

Using Verus-Desktop

1. download the new version
2. extract the archive
3. verify the signature using the data in the `*.signature.txt`-file through your existing Verus-Desktop, *VerusID* tab, Verify Signed Data and choose to verify a file. Only continue when this verification returns True.
4. stop your Verus-Desktop wallet
5. install the verified installer (Windows).
6. start your wallet.

Using CLI Wallet

1. download the new version
2. extract the archive

3. verify the signature using the data in the `*.signature.txt`-file with the command `./verus verifyfile "address or identity" "signature" "filepath/filename"` command. Only continue when this verification returns True.
4. stop your verusdaemon verus stop
5. extract the verified archive to your current CLI-wallet location
6. start your wallet (verusd)

How-To: Backup my wallet?

Important General Information

Video:

Watch this video with an explanation how to accomplish the steps below: Backup your wallet

Verus `Wallet.dat`, Chaindata & `VRSC.conf` standard locations

- Linux: `~/.komodo/VRSC`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`
- OS independent through Verus Desktop: Click `help`, `Show Verus data folder (default)`

Instruction Video

External YouTube link: Backing up the Verus Desktop wallet

Preferred method: Exporting your wallet

Note: The filename you replace`<mywalletexport>` with, can only contain letters and figures, no other characters, so it **cannot** have a file-extension!*

Verus Desktop:

Go to `Settings`, `Coin Settings` and click `Export native wallet backup`.
Confirm that you want to export after reading the pop-up.
The green message will tell you where the backup is and what it's name is.

linux/MacOS CLI:

run `./verus z_exportwallet "<mywalletexport>"`

windows CLI:

run `verus z_exportwallet "<mywalletexport>"`

Attention: Pay attention to the feedback this command gives you: it will mention the location where the export file is saved.

The exported wallet should be a file called `<mywalletexport>`, standard in the same directory as your `wallet.dat`. Keep this file secure, it has your plaintext private keys. Verify that the file is there and isn't empty.

Alternate method: Backing up your wallet

Note: The filename you replace`<DestinationFileName>` with, can only contain letters and figures, no other characters, so it **cannot** have an file-extension

Verus Desktop:

Go to `Settings`, `Coin Settings` and click the text box
type `run backupwallet "<DestinationFileName>"`

linux/MacOS CLI:

run `./verus backupwallet "<DestinationFileName>"`

windows CLI:

run `verus backupwallet "<DestinationFileName>"`

Attention: Pay attention to the feedback this command gives you: it will mention the location where the backup file is saved.

The backup wallet should be a file called `<DestinationFileName>`, standard in the same directory as your `wallet.dat`. Keep this file secure, it enables full access to all your addresses. Verify that the file is present and that it is the same size as your `wallet.dat`.

Extra info for non-Verus chains

Extra line in `<coin>.conf` required

Non-Verus chains like **Komodo** and its asset chains and **Zcash**, need this entry in the coins configuration file to specify the export directory, before you started your wallet.
`exportdir=<dir>`

For Komodo the base directory is `komodo`.

- Linux: `~/.Komodo`
- Mac OS: `~/Library/Application Support/Komodo`
- Windows 10: `%AppData%\Roaming\Komodo`
  For Komodo asset chains it is a folder/directory in the `komodo` base directory (eg `komodo/PIRATE`) with the **official** coin designation.

For Zcash the base directory is `zcash` instead of komodo.

- Linux: `~/.Zcash`

- Mac OS: `~/Library/Application Support/Zcash`
- Windows 10: `%AppData%\Roaming\Zcash`

non-Verus chains Verus Desktop

- **before** executing the command in the `run ...` commands, select the appropriate coin in the top right corner.

How-To: Restore my wallet from a backup?

Important General Information

Verus `Wallet.dat`, Chaindata & `VRSC.conf` standard locations

- Linux: `~/.komodo/VRSC`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`
- OS independent through Verus Desktop: Click `help`, `Show Verus data folder (default)`

`verus command "<userinput>"` needs to be entered literally, with `<userinput>` replaced by your specific userdata. So if the text directs you to use for example `"<Public Address>"`, you replace that (including the < and >) with the address,
so it looks similar to this: `"RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r"`.

Instruction Video

External YouTube link: Restoring the Verus Desktop wallet

Procedure

Using a backup of your `wallet.dat`.

1. Stop verusd. For Windows-Desktop or Agama, just exit and wait for it to close completely. For the linux cli run `./verus stop`, or for the windows cli run `verus stop`.
2. Once your wallet is finished closing copy the backup of your `wallet.dat` file from your backup location to the directory listed in the start of this document (see above).
3. Now restart your wallet by launching Verus Desktop, Agama or running verusd for the CLI.

Using a `walletexport` file.

Note: The filename you replace`<mywalletimport>` with, can only contain letters and figures, no other characters, so it **cannot** have an file-extension

Attention: The command `z_importwallet` triggers the wallet to rescan in order to make all transactions to the freshly imported wallet addresses visible.
Rescanning your wallet may take a considerable time, during which your wallet may not respond to other commands. Please be patient.

The `<PATH>` in the `z_importwallet` command needs to be the **full absolute** path to the file. replace `LOGINNAME` with the actual loginname.

Verus Desktop:

Go to `Settings`, `Coin Settings` and click `Import native wallet backup`.
Click `Choose file`, browse to your backup file, select it and click `Open`
Click `Import` to start the inport process

Agama:

Go to settings, scroll to the bottom and click CLI, select VRSC in that section.
Then below type `z_importwallet "<PATH><mywalletimport>"` and click the button below to run it.

linux/MacOS CLI:

run `./verus z_importwallet "<PATH><mywalletimport>"`

Windows CLI:

run `verus z_importwallet \"<PATH><mywalletimport>\"`

Using individual seeds / WIF-keys

Importing individual keys is explained in detail in this wiki: [import your Lite wallet address into your native Verus Desktop].

Information compiled by Oink.vrsc@.

Guide to change Verus-Desktop from Lite Mode to Native Mode.

Attention: Read completely before use.

Important General Information

Verus `Wallet.dat`, Chaindata & `VRSC.conf` standard locations

- Linux: `~/.komodo/VRSC`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`

- OS independent through Verus Desktop: Click `help`, `Show Verus data folder (default)`

Necessary files & links:

Link 1: Download Verus Bootstrap
Link 2: Import Lite wallet address in Verus Desktop native
Link 3: Checking the signature

Procedure:

1. Make sure you have your seed phrase and password you use to log into your Lite mode wallet available.
2. First of all make a notition of your address and balance of VRSC you have in your wallet, before closing Verus Desktop.
3. Make sure the latest version of Verus-Desktop is installed.
    1. Download the latest Verus-Desktop.
    2. Verify the signature of your download, so you have an untampered installer. Link 3 or Video
    3. Run the file you just downloaded to install it.
4. Installing the bootstrap:
    1. Download the bootstrap from Link 1.
    2. Verify the SHA256 checksum of your download, to verify you have an untampered Bootstrap.
    3. Unpack the downloaded archive to `VRSC wallet & data location`.
5. Getting Verus-Desktop ready for Native mode:
    1. Start Verus-Desktop and enter your profile (if not loaded automatically).
        1. If present in your profile, `deactivate` Verus Lite.
        2. Click `+ Add Coin`, select **Verus** from the dropdown list and continue.
        3. Select **Native** and optionallu tick the options `Start staking`, `Start Mining` and fill in the amount of threads to mine with.
        4. Click `Add Coin`. Verus-Desktop will add Verus as Native chain to your screens.
    2. You may get a red warning message about Zcash params. (Verus Desktop will detect if you have the necessary ZCash parameter files and download them if needed)
    3. As soon as the download is finished, Verus-Desktop will continu and bring you into your wallet. It will automatically start to synchronize the blockchain. Since we already put the majority of the chain in place, this will take just a few minutes.
6. Importing your existing Address:
    1. This procedure is described in detail in: Import Lite wallet address in Verus Desktop native.

If you followed these steps, installed the bootstrap, switched from Lite to Native mode and imported your existing address into Verus-Desktop. You can now stake your balance and use Private (sapling) addresses.

Created by Oink.vrsc@

How to import your Lite wallet address into your native Verus Desktop?

Important General Information

`verus command "<userinput>"` needs to be entered literally, with `<userinput>` replaced by your specific userdata. So if the text directs you to use for example `"<Public`

`Address>"`, you replace that (including the `<` and `>`) with the address,
so it looks similar to this: `"RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r"`.

This method is confirmed to work on Verus and all Komodo assetchains.

Prerequisites

In order to convert your seedphrase into a Private key (WIF), you need to have a running native wallet first, that is fully synchronized. The earliest wallet that supports these functions is **Verus Desktop v0.6.4-beta-1**.

If needed, use this guide to quickly synchronize your wallet: https://wiki.verus.io/#!how-to/how-to_bootstrap.md

Converting Seed to WIF

If you have a seed, you can retrieve your Private key (WIF) by having the Verus Desktop wallet convert it for you.
To convert your *seed phrase* in Verus Desktop, go to `settings` --> `Coin Settings`, select the chain you want to use in the top-right corner and enter the following command:

```
run convertpassphrase "word_1 word_2 word_3 ... word_n"
```

Note: Make sure you replace `Word1 word2 word3 ... word_n` with the actual seedphrase (12 or 24 words) of the address you want to import!

You will receive a response **similar** to this:

```
{
"walletpassphrase": "seedphrase",
"address": "RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r",
"pubkey": "02ffc2f4b071afdec631e3fb7d435a0047be14a81ea1a269e4206b0068c0c1fa6f",
"privkey": "d899ed88e9ee2e90c2cf51cb47e7b4495ec1e1cb10763bb1c111b0bde48bf86c",
"wif": "UwGb5KvGPfMUr1tu74Desjh87ZeJM4wq5goLyThcogeLifc5aJqT"
}
```

Copy that information and store it somewhere **SAFE**. With this information anyone having access to it will have full control over that address.

The 52-character string after **`"wif":`** that is shown, is what you want to import in the next step.

Importing a single WIF for a public (transparent) address

To import your address, go to `settings` --> `Coin Settings`, select the chain you want to use in the top-right corner and enter the following command:

```
run importprivkey "<wif>" "" true
```

Replace `<wif>` with the actual **wif** you got from the `convertpassphrase` command earlier.

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

Note: The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Importing multiple WIFs in one batch for public (transparent) addresses

To import your address, go to `settings` --> `Coin Settings`, select the chain you want to use in the top-right corner and enter the following command for every WIF except for the final one:

```
run importprivkey "<wif>" "" false
```

Import the final WIF with this command:

```
run importprivkey "<wif>" "" true
```

The last command triggers the chain to rescan all addresses in your wallet, including all the addresses you just imported.
Replace `<wif>` with the actual **wif** (like the one you got from the `convertpassphrase` command earlier).

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Importing a single WIF for a private address

To import your address, go to `settings` --> `Coin Settings`, select the chain you want to use in the top-right corner and enter the following command:

```
run z_importkey "<wif>" "yes" 1
```

Replace `<wif>` with the actual **wif** you got from the `convertpassphrase` command earlier.

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Importing multiple WIFs in one batch for private addresses

To import your address, go to `settings` --> `Coin Settings`, select the chain you want to use in the top-right corner and enter the following command for every WIF except for the final one:

```
run z_importkey "<wif>" "no"
```

Import the final WIF with this command:

```
run z_importkey "<wif>" "yes" 1
```

The last command triggers the chain to rescan all addresses in your wallet, including all the addresses you just imported.
Replace `<wif>` with the actual **wif** (like the one you got from the `convertpassphrase` command earlier).

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

How do I install the Verus CLI (command line interface) wallet on a brand new (hosted) Linux system?

Notice: Read it completely before use.

Important General Information

`wallet.dat` location on Linux: `~/.komodo/VRSC`

Necessary files:

Link 1: Download latest Wallet
Link 2: Download Verus Bootstrap

Procedure:

1. First make sure your system is up to date:
   ```
   sudo apt-get update && apt-get upgrade -y
   ```
2. I suggest not to use the root account. I you have not yet done, set up a new user
   ```
   sudo adduser newusername
   sudo usermod -aG sudo newusername
   ```
3. Switch to new username
   ```
   su - newusername
   ```
4. Test the your new user actually has root (SuDo) access, e.g.:
   ```
   sudo ls -la /root
   ```

You should get some lines like this:

```
drwx------ 6 root root 4096 Jul 3 15:56
```

5. Download & install the wallet binaries:

```
wget
https://github.com/VerusCoin/VerusCoin/releases/download/v0.9.
3/Verus-CLI-Linux-v0.9.3-amd64.tgz
```

The downloaded archive contains another archive and a signature text file, enabling the archive within to be verified (You'll need a running wallet to do that)

Also: Verify the URL to the latest version from the [Download latest Wallet] above.

```
tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tgz
```

Now extract the wallet archive:

```
tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tar.gz
```

Change directory to verus-cli

```
cd verus-cli
```

Fetch parameters, takes time, more on slow Internet connection

```
./fetch-params
```

Creating the chaindata directory

```
cd ~
mkdir -p .komodo/VRSC
cd ~/.komodo/VRSC
```

Download the block-chain bootstrap, this considerably speeds up synchronisation of the block-chain from days to minutes... (optional)

```
wget https://bootstrap.verus.io/VRSC-bootstrap.tar.gz
tar -xvf VRSC-bootstrap.tar.gz
```

Install libraries for Verus

```
sudo apt-get install libcurl3 g++-multilib -y
```

Install Tmux a terminal multiplexer with which you can run threads in the background see https://en.wikipedia.org/wiki/Tmux

```
sudo apt-get install tmux -y
```

Start tmux:

```
tmux
```

Launch Verus Daemon with or without number of threads

(usually number of threads equals number of cores or double of that if the processor support hyper threading well)

```
~/verus-cli/verusd -gen -genproclimit
~/verus-cli/verusd -gen -genproclimit=24
```

Once mining is operational – again this may take some time –

you’ll see: 256 mega hashes complete - working

then detach tmux

```
[ctrl]&b d
```

Disable login with Root User

(make sure your newly created user login works and has sudo rights)

```
sudo nano /etc/ssh/sshd_config
```

Find: PermitRootLogin yes

And set to

PermitRootLogin no
Apply new settings:
```
sudo systemctl restart sshd
```

How do I install the Verus CLI (command line interface) wallet on a brand new (hosted) Linux system?

Notice: Read it completely before use.

Important General Information

`wallet.dat` location on Linux: `~/.komodo/VRSC`

Necessary files:

Link 1: Download latest Wallet
Link 2: Download Verus Bootstrap

Procedure:

1. First make sure your system is up to date:
   ```
   sudo apt-get update && apt-get upgrade -y
   ```
2. I suggest not to use the root account. I you have not yet done, set up a new user
   ```
   sudo adduser newusername
   sudo usermod -aG sudo newusername
   ```
3. Switch to new username
   ```
   su - newusername
   ```
4. Test the your new user actually has root (SuDo) access, e.g.:
   ```
   sudo ls -la /root
   ```
   You should get some lines like this:
   ```
   drwx------ 6 root root 4096 Jul 3 15:56
   ```
5. Download & install the wallet binaries:
   ```
   wget
   https://github.com/VerusCoin/VerusCoin/releases/download/v0.9.
   3/Verus-CLI-Linux-v0.9.3-amd64.tgz
   ```
   The downloaded archive contains another archive and a signature text file, enabling the archive within to be verified (You'll need a running wallet to do that)
   Also: Verify the URL to the latest version from the Download latest Wallet above.
   ```
   tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tgz
   ```

Now extract the wallet archive:

```
tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tar.gz
```

Change directory to verus-cli

```
cd verus-cli
```

Fetch parameters, takes time, more on slow Internet connection

```
./fetch-params
```

Creating the chaindata directory

```
cd ~
mkdir -p .komodo/VRSC
cd ~/.komodo/VRSC
```

Download the block-chain bootstrap, this considerably speeds up synchronisation of the block-chain from days to minutes... (optional)

```
wget https://bootstrap.verus.io/VRSC-bootstrap.tar.gz
tar -xvf VRSC-bootstrap.tar.gz
```

Install libraries for Verus

```
sudo apt-get install libcurl3 g++-multilib -y
```

Install Tmux a terminal multiplexer with which you can run threads in the background see https://en.wikipedia.org/wiki/Tmux

```
sudo apt-get install tmux -y
```

Start tmux:

```
tmux
```

Launch Verus Daemon with or without number of threads
(usually number of threads equals number of cores or double of that if the processor support hyper threading well)

```
~/verus-cli/verusd -gen -genproclimit
~/verus-cli/verusd -gen -genproclimit=24
```

Once mining is operational – again this may take some time –
you'll see: 256 mega hashes complete - working
then detach tmux

```
[ctrl]&b d
```

Disable login with Root User
(make sure your newly created user login works and has sudo rights)

```
sudo nano /etc/ssh/sshd_config
```

Find: PermitRootLogin yes
And set to
PermitRootLogin no
Apply new settings:

```
sudo systemctl restart sshd
```

How do I install the Verus CLI (command line interface) wallet on a brand new (hosted) Linux system?

Notice: Read it completely before use.

Important General Information

`wallet.dat` location on Linux: `~/.komodo/VRSC`

Necessary files:

Link 1: Download latest Wallet
Link 2: Download Verus Bootstrap

Procedure:

1. First make sure your system is up to date:
   ```
   sudo apt-get update && apt-get upgrade -y
   ```
2. I suggest not to use the root account. I you have not yet done, set up a new user
   ```
   sudo adduser newusername
   sudo usermod -aG sudo newusername
   ```
3. Switch to new username
   ```
   su - newusername
   ```
4. Test the your new user actually has root (SuDo) access, e.g.:
   ```
   sudo ls -la /root
   ```
   You should get some lines like this:
   ```
   drwx------ 6 root root 4096 Jul 3 15:56
   ```
5. Download & install the wallet binaries:
   ```
   wget
   https://github.com/VerusCoin/VerusCoin/releases/download/v0.9.
   3/Verus-CLI-Linux-v0.9.3-amd64.tgz
   ```
   The downloaded archive contains another archive and a signature text file, enabling the archive within to be verified (You'll need a running wallet to do that)
   Also: Verify the URL to the latest version from the Download latest Wallet above.
   ```
   tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tgz
   ```
   Now extract the wallet archive:
   ```
   tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tar.gz
   ```
   Change directory to verus-cli
   ```
   cd verus-cli
   ```
   Fetch parameters, takes time, more on slow Internet connection
   ```
   ./fetch-params
   ```

Creating the chaindata directory

```
cd ~
mkdir -p .komodo/VRSC
cd ~/.komodo/VRSC
```

Download the block-chain bootstrap, this considerably speeds up synchronisation of the block-chain from days to minutes... (optional)

```
wget https://bootstrap.verus.io/VRSC-bootstrap.tar.gz
tar -xvf VRSC-bootstrap.tar.gz
```

Install libraries for Verus

```
sudo apt-get install libcurl3 g++-multilib -y
```

Install Tmux a terminal multiplexer with which you can run threads in the background see https://en.wikipedia.org/wiki/Tmux

```
sudo apt-get install tmux -y
```

Start tmux:

```
tmux
```

Launch Verus Daemon with or without number of threads
(usually number of threads equals number of cores or double of that if the processor support hyper threading well)

```
~/verus-cli/verusd -gen -genproclimit
~/verus-cli/verusd -gen -genproclimit=24
```

Once mining is operational – again this may take some time –
you'll see: 256 mega hashes complete - working
then detach tmux

```
[ctrl]&b d
```

Disable login with Root User
(make sure your newly created user login works and has sudo rights)

```
sudo nano /etc/ssh/sshd_config
```

Find: PermitRootLogin yes
And set to
PermitRootLogin no
Apply new settings:

```
sudo systemctl restart sshd
```

Shielding Verus Coin via the CLI.

Attention: Read it completely before use.

Note: Shielding is no longer required for coinbase rewards after block 800200. Earlier timelocked coins will still need to be shielded in order to use them.

Important General Information

`verus command "<userinput>"` needs to be entered literally, with `<userinput>` replaced by your specific userdata. So if the text directs you to use for example `"<Public Address>"`, you replace that (including the `<` and `>`) with the address,
so it looks similar to this: `"RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r"`.

General remarks on CLI wallet:

On Windows command line enter the commands as shown without the surrounding quotation marks
In Linux shell preceed the commands without surrounding quotation marks with `./`
In MacOS shell preceed the commands without surrounding quotation marks with `./`
Example: the windows version `verus listtransactions` transforms in Linux or MacOS to `./verus listtransactions`.

General remarks on Windows command line formatting:

The CLI help shows the command format for Linux and MacOS.
For windows substitute the shown `'`-character with the `"`-character.
For windows substitute the shown `"`-character with the `\"`-characters.

Procedure:

You must first "shield coins" (send from a transparent R-addr to a shielded zaddr) to be able to use them in staking, when they first unlock.
This is part of the Zcash protocol itself. The commands below assume you are in the Verus source code directory

```
verus z_shieldcoinbase
```

You can either use an existing zaddr or use a new zaddr. To make a new zaddr:

```
verus z_getnewaddress
```

Use the address the above command outputs in the `z_shieldcoinbase` command

To shield all coinbase in your wallet, you can use `"*"` (quotes are important) and zaddr that is getting the funds:

```
verus z_shieldcoinbase "*" <YOUR zs-ADDRESS>
```

To just shield a single address, specify that as the first argument:

```
verus z_shieldcoinbase <YOUR R-ADDRESS> <YOUR zs-ADDRESS>
```

Once the funds have moved to the zaddr and are confirmed, you can freely send them to any address, They will be eligble for staking id sent to a transpartent address (R-address).

To send from a certain zaddr to a transparent address, use z_sendmany. The following command sends 10 Verus to a given transparent address, ID address or ID-name.
Example: `verus z_sendmany zcZpfuzzJqmNJ3fUJekvbnyuxuJe9eAURAHrMCvN2Nr7VuWjakb1LEw6j2etPcCnr4`

```
5BRot7MaMbipuS5da162BfuUkFGxx '[{"amount":10,"address":"Verus Coin
Foundation@"}]'
```

Shielding Verus Coin via the CLI.

Attention: Read it completely before use.

Note: Shielding is no longer required for coinbase rewards after block 800200. Earlier timelocked coins will still need to be shielded in order to use them.

Important General Information

`verus command "<userinput>"` needs to be entered literally, with `<userinput>` replaced by your specific userdata. So if the text directs you to use for example `"<Public Address>"`, you replace that (including the `<` and `>`) with the address,
so it looks similar to this: `"RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r"`.

General remarks on CLI wallet:

On Windows command line enter the commands as shown without the surrounding quotation marks
In Linux shell preceed the commands without surrounding quotation marks with `./`
In MacOS shell preceed the commands without surrounding quotation marks with `./`
Example: the windows version `verus listtransactions` transforms in Linux or MacOS to `./verus listtransactions`.

General remarks on Windows command line formatting:

The CLI help shows the command format for Linux and MacOS.
For windows substitute the shown `'`-character with the `"`-character.
For windows substitute the shown `"`-character with the `\"`-characters.

Procedure:

You must first "shield coins" (send from a transparent R-addr to a shielded zaddr) to be able to use them in staking, when they first unlock.
This is part of the Zcash protocol itself. The commands below assume you are in the Verus source code directory

```
verus z_shieldcoinbase
```

You can either use an existing zaddr or use a new zaddr. To make a new zaddr:

```
verus z_getnewaddress
```

Use the address the above command outputs in the `z_shieldcoinbase` command

To shield all coinbase in your wallet, you can use `"*"` (quotes are important) and zaddr that is getting the funds:

```
verus z_shieldcoinbase "*" <YOUR zs-ADDRESS>
```

To just shield a single address, specify that as the first argument:

```
verus z_shieldcoinbase <YOUR R-ADDRESS> <YOUR zs-ADDRESS>
```

Once the funds have moved to the zaddr and are confirmed, you can freely send them to any address, They will be eligble for staking id sent to a transpartent address (R-address).

To send from a certain zaddr to a transparent address, use z_sendmany. The following command sends 10 Verus to a given transparent address, ID address or ID-name.
Example: `verus z_sendmany zcZpfuzzJqmNJ3fUJekvbnyuxuJe9eAURAHrMCvN2Nr7VuWjakb1LEw6j2etPcCnr45BRot7MaMbipuS5da162BfuUkFGxx '[{"amount":10,"address":"Verus Coin Foundation@"}]'`

Shortly after you confirmed all and it worked, you should see a "Commit Name" window pop up. This window will confirm that the process of
checking and registering your ID name has successfully started.
You can close this window by clicking on "Done".
When the "Commit Name" window is closed, you will see the status of the name checking and reservation on the top right of your Identity
Overview window.
As soon as the status changes from a blue "Pending" to a green "Ready", we can finally go on to register your ID.
Please note: You haven't purchased your ID yet!
Until now, we only checked and reserved a name, and we set the basics that are necessary for that. The ID is not yet registered!
You can also see this in your transactions, where you will see one outgoing and one incoming transaction of 0 VRSC from and to the public
address you have chosen above. The process of checking and reservation is for free (except for one transaction fee).
So, until this point, you can still cancel the process by clicking the grey trash can with an X on it, next to that. If you don’t want to cancel, let’s
continue.
Now, we will finally register your ID.
Please click on "Create Verus ID", which is right to the „Ready“ status note in your Identity Overview.
You will see a window pop up with three fields to fill.
These fields are:
"Enter revocation ID": If you have another ID, or a friend, or family member with an ID you really trust**, you can enter it here. This ID will be
able to revoke your ID. It's a safety in case your ID gets compromised in any way, or if you can't access your wallet any more, or similar.
If you don’t know of any ID you want to enter here, you can leave it blank. If you want to make the ID independent and fully self-
sovereign, you can click the "SELF" button. But you may also leave that line empty..
"Enter recovery ID": This ID will allow you to recover a revoked ID **, to make it valid again. Same rules as above, if you have somebody
with an ID you trust, or you already have another ID, you can use it. For self-sovereign

IDs, use the “SELF” button. And you can of course
leave it empty, too.**
"Enter private address": Ok, that's needed to receive private transaction. Usually we don't have that memorised, so we need to go back to
the wallet and copy the address.
**) Some notes on IDs
You see that you can enter three IDs, viz. the main ID, the revocation ID, and the recovery ID. Actually they are equal for their own purpose,
each of them is an own, normal, independent ID, and can do anything the main ID can do on their own. They don’t even have to be your IDs.
But they just can be assigned to different roles (a.k.a. authority). If you choose to do so, here’s what can be controlled by them:
The main ID is the one you would do all transactions with, and other stuff that is still to come. It always has the power to change both of
the other IDs.
The Revocation ID can revoke the main ID, but also has control over changing anything that is related to the Revocation ID itself. This
means, if you change the Revocation ID to an ID you do not control, you no longer can change the Revocation ID again, except from the
wallet that also controls the Revocation ID. This also applies to Recovery ID.
The Recovery ID can of course recover the main ID when it was revoked. And (only) in this process, it can also change the public and
private addresses of the main ID, and it’s Revocation ID.
Both the Revocation ID and the Recovery ID can be used for more than one main ID, so you only need to create them once. And they can
be the same, too, so you can use one other ID both for revocation and recovery.
If you want to be able to revoke and recover your main ID, you need at least one more ID for that. The main ID can not revoke itself.
However, if you want to have a totally self-sovereign ID, and control the private and public address it is assigned to in another way, you
can use self-assignment.
Revocation and Recovery ID(s) should be created before your main ID. Although there is a way to change them later, it’s rather
complicated (at time of writing this guide). So if you plan to be able to revoke and recover, it’s much easier to create them first.
Now you see what these IDs can control. What they do not control, nor revoke or recover, are the underlaying public (R-) and private (z-)
addresses that you enter on creation. These will always be in control of the wallet that has the private keys (a.k.a. WIFs) for them. They will
stay unchanged, can’t neither be revoked nor recovered. That is only possible for IDs.
You will, however, find a new address after creating an ID, which is an i-address. This address is tied to an ID, and only exists together with a
valid ID. It does not have a private key, and can not be restored other than by the associated ID.
For the experienced user it may even be an option to create an ID (e.g., for revocation or recovery) on a new wallet, with a separate
wallet.dat file. The wallet.dat can be saved and secured offline, so it is at hand when needed. But you only should do so if you’re really sure
what you are doing. If you would like to have some support with this, feel free to ask in our Discord https://discord.gg/VRKMP2S.
Now, after you know a bit more about IDs, you can decide how many and which IDs you want to create.

Let’s go on in creating your first ID, go back to the Wallet tab (on the left, above the blue ribbon).
There, when you click on the Verus badge, you can see your funds, both the transparent (i.e., public), and the private balance.
To copy a private address, you simply click on the green down-arrow in the Private Balance badge.
If you don't have a private address yet, or you don't want to use one of those you have, now simply create a new private (z-) address by
clicking the blue button.
To copy the private address then just click the "Copy" button next to your private address.
Now you can go back to the "Verus Identities" tab and click again on "Create Verus ID" for your reserved name. As you don’t have a
revocation or recovery ID yet we will leave that empty for now. Just paste your private address in the last field and click "Continue".
Now a confirmation window will pop up that shows all the information you entered. It will also show the fee that will be deducted for
purchasing, and if you entered a Referral ID, the i-address of which is also shown.
If you want to cancel the creation of your ID, this will be the last possibility to do that.
However, if you want to create the ID you can see there, click on „Continue“.
You may now see a window popping up shortly for the process.
But right after that, the confirmation window will show again, together with a blue ribbon below it to confirm that your 100 (or 80) Verus
coins were transferred to purchase your ID.
All you have to do now is wait a little, until the transaction was confirmed by the network.
You may notice that the fee was deducted from
your funds, also visible in transactions, and your wallet shows the remaining funds.
And then - right after one confirmation of the network - your ID is ready to use. You can see that in the Verus ID tab:
Congratulations, you are now proud owner of a Verus ID!
Finally, a few notes on referrals.
When you enter a referral ID, purchasing an ID will cost 80 instead of 100 VRSC (not including transaction fees). At the same time, the referral
ID will get 20 VRSC.
Furthermore, when someone is using your ID as a referral, you will get 20, and the referral you were using will again get 20 VRSC (but
purchase will not get cheaper than 80 VRSC).
This referral propagation will work up to the 3rd generation, i.e., up to the referrer of the referrer of the referrer. Each of them will have 20
VRSC. But at least 20 VRSC will always go back to the miners and stakers of Verus Coin, and will be added to one of the coming blocks. That’s
also why mining and staking will become more profitable with each ID that was created.
Thanks for reading, and don’t hesitate to ask in our discord if you need support! :)
veruscoin.io

How to create your first Verus ID with CLI -
This guide will explain how to get your first Verus ID using the Command Line Interface. I recommend reading it all down first, and then go
through the process of creating an ID.
If you don't the wallet installed yet, check our website at https://veruscoin.io/wallet.html. Scroll down to Wallets and choose your operating
system.
If you didn't set up your wallet before, please also refer to the guides on that page,

especially see the "FAQ and How-to" section at
https://veruscoin.io/service.html, and check the Bootstrap procedure (marked in above image) to speed up syncing of your wallet at
https://veruscoin.io/downloads/how-to/how-to_bootstrap.html
If you have troubles, please visit us on Discord https://discord.gg/VRKMP2S and ask in the #community-support channel, so somebody can
support you.
Ok, back to topic. This guide will explain how you can get your first ID with a clean and fresh installed CLI wallet. Some of the steps will no
longer apply when you already have a working version of that wallet, but you will notice.
Getting a Verus ID requires a balance of 100.0002 VRSC in your wallet (or 80.0002 if you have a referral ID). The ID will cost 100 (or 80) VRSC,
and there is a transaction fee of 2 x 0.0001 VRSC (at time of writing this guide). So, if you don't have that much, you might want to get that
before. If you don’t know how, please also see https://veruscoin.io/getVRSC.html for that.
When you want to create an ID:
🕐 start the CLI daemon (The example below will run as a daemon and also stake your balance and mine on 1 thread)
```
./verusd -mint -gen -genproclimit=1 -daemon 1>/dev/null 2>&1
```
🕐 Once the daemon is started, check your blockheight and compare it to the current blockheight on https://explorer.veruscoin.io
```
./verus checkblockcount
```
which should result in a blocknumber:
```
838996
```
🕐 Now it is time to check if you have sufficient funds to acquire an ID. Check your balance with:
```
./verus z_gettotalbalance
```
🕐 which should show an output similar to this:
```
{
"transparent": "640.0310769",
"interest": "0.00",
"private": "0.00",
"total": "640.0310769"
}
```
🕐 If it's less than 100, you need to get some more before you can create an ID. If you have recently sent funds to your address and that
isn’t included in this balance, you can wait for the transfer to show up.
🕐 To create an ID, you have to check first if that ID already exists with this command (I’ll be using Demo ID in this guide):
```
./verus getidentity "Demo ID@"
```
If the identity is not yet registered you get the following output:
```
error code: -5
error message:
Identity not found
```
If the identity is already registered, you will be provided with details about that ID
🕐 You can use any character except \ / : . * ? " < > | @ for your ID
◦ And you can not use an empty space as a first character.
◦ But don't worry, the system won't let you commit such names, even if you try.
◦ What you can use is e.g. an empty space in between, which would allow you to commit a name like "Lucky Joe" (without the
quotes of course), but also "Lucky_Joe", "LuckyJoe", and even "Lüçkÿ Jöë" or "幸运 ノ\ノ\

/\" would work. There is no rule for
creating a useful ID, and you can create several IDs if you like. That means, you can create a matching ID for each purpose.

🕐 The command to commit a name for your ID:

```
registernamecommitment "Demo ID"
"RFoinkqD2QinMi25mUu5h7k3f4ETaBQWpo" "Verus Coin Foundation@"
```

◦ The text between the first pair of quotes is your ID name
◦ The text between the second pair of quotes is the address in your wallet that pays the fees to create this ID
◦ The text between the third pair of quotes is the optional referral ID, which lowers your purchase price.

The command above will return you a json formatted message:


```
{
"txid":
"862485a6af4b6989531cf530e5a13b531243c805702de40b0d8086f3527c174c"
,
"namereservation": {
"name": "Demo ID",
"salt":
"739e248f525c8fbbb01c4c480a6c325c10e28ab1a00dc0f81b6583e1b9cab261"
,
"referral": "i5v3h9FWVdRFbNHU7DfcpGykQjRaHtMqu7",
"parent": "",
"nameid": "iRLC6sRkSACDVRYEoz32iY6kE9ax5xyTyE"
}
}
```


🕐 The message that the command above returned, contains the information which is needed for the registration of the ID, but the
above transaction first has to be mined into the blockchain, so you will have to wait at least 1 block before you can issue the next
command. To make sure the transaction is mined into the chain you can check the TXID from that message on the explorer:
https://explorer.veruscoin.io

🕐 The command to register your ID:

```
./verus registeridentity '{"txid":
"862485a6af4b6989531cf530e5a13b531243c805702de40b0d8086f3527c174c"
, "namereservation": { "name":
"Demo ID", "salt":
"739e248f525c8fbbb01c4c480a6c325c10e28ab1a00dc0f81b6583e1b9cab261"
, "referral":
"i5v3h9FWVdRFbNHU7DfcpGykQjRaHtMqu7", "parent": "", "nameid":
"iRLC6sRkSACDVRYEoz32iY6kE9ax5xyTyE"}, "identity":{"name":"Demo ID",
"primaryaddresses":["RFoinkqD2QinMi25mUu5h7k3f4ETaBQWpo"],
"minimumsignatures":1,
"privateaddress":
"zs16xdne62g2wl62vq8kmwl03klv4qs5zf3jxmax9vj7f443ymj59ds4qtvj8tp67
zvs58fvvpc22n"}}'
```

◦ The data shown in red is copied from the response message from the commit command.
◦ The data shown in green is your public address that your ID is bound to.
◦ The data shown in purple, is your private address, which can be used for private transfers, messaging, voting, polling etcetera.

🕐 After running this command the CLI will respond with a single line containing the TXID:
`de2909a8832a94182037f3ea29f7382b5a2841f505ab20ceb8334e3712127078`
🕐 Once this TXID is mined into a block on the blockchain (again: you can use the explorer to check), your ID is registered. You can check the registration using:
`./verus listidentities`
The output will show you all identities in your wallet:

```
[
{
"identity": {
"version": 1,
"flags": 0,
"primaryaddresses": [
"RFoinkqD2QinMi25mUu5h7k3f4ETaBQWpo"
],
"minimumsignatures": 1,
"identityaddress": "iNcmBb1z1pdW2qW4UDhNp1MXg7Rq9Yk8ge",
"parent": "RQVsJRf98iq8YmRQdehzRcbLGHEx6YfjdH",
"name": "!",
"contentmap": {
},
"revocationauthority": "iNcmBb1z1pdW2qW4UDhNp1MXg7Rq9Yk8ge",
"recoveryauthority": "iNcmBb1z1pdW2qW4UDhNp1MXg7Rq9Yk8ge",
"privateaddress":
"zs1sx3yklu4ezq8ht5xnghrjhm4jxs8mcgxaqlky50e548mqngv8u8vnptrluq5w9
jft2ackpxsh9e"
},
"blockheight": 3701,
"txid":
"48ba2a4e1a5e5ee0cada495b268b85679fc43bf573a3ad86aa1e117e37f1b848"
,
"status": "active",
"canspendfor": true,
"cansignfor": true
}
]
```

The procedure above creates IDs that have all authorities set to themselves.
🕐 If you want to update your ID after creating, your can use the command updateidentity. In our example it would look like this, if I wanted to update the revocation authority and recovery authority to other existing Identities:

```
./verus updateidentity '{"version": 1, "flags": 0, "primaryaddresses":
["RFoinkqD2QinMi25mUu5h7k3f4ETaBQWpo"], "minimumsignatures": 1,
"identityaddress":
"iNcmBb1z1pdW2qW4UDhNp1MXg7Rq9Yk8ge", "parent":
"RQVsJRf98iq8YmRQdehzRcbLGHEx6YfjdH", "name": "Demo ID",
"contenthashes": [], "revocationauthority":
"i5vt1Cx7oDMjfFSNSFpdMFQLoyD9gqDN55", "recoveryauthority":
"iLbcafWjFgvrZ9yGNKofWmBw1DgJyAaVe2", "privateaddress":
"zs1sx3yklu4ezq8ht5xnghrjhm4jxs8mcgxaqlky50e548mqngv8u8vnptrluq5w9
jft2ackpxsh9e"}'
```

🕐 The ID addresses in red reflect the changes I made, to assign the authorities to other

IDs.
🕐 The "revocationauthority": "i5vt1Cx7oDMjfFSNSFpdMFQLoyD9gqDN55" and "recoveryauthority": "iLbcafWjFgvrZ9yGNKofWmBw1DgJyAaVe2" can also added to the registration process, to assign these authorities right away. The
resulting command would look like this, in our example:

```
./verus registeridentity '{"txid":
"862485a6af4b6989531cf530e5a13b531243c805702de40b0d8086f3527c174c"
, "namereservation": { "name":
"Demo ID", "salt":
"739e248f525c8fbbb01c4c480a6c325c10e28ab1a00dc0f81b6583e1b9cab261"
, "referral":
"i5v3h9FWVdRFbNHU7DfcpGykQjRaHtMqu7", "parent": "", "nameid":
"iRLC6sRkSACDVRYEoz32iY6kE9ax5xyTyE"}, "identity":{"name":"Demo ID",
"primaryaddresses":["RFoinkqD2QinMi25mUu5h7k3f4ETaBQWpo"],
"minimumsignatures":1,
"privateaddress":
"zs16xdne62g2wl62vq8kmwl03klv4qs5zf3jxmax9vj7f443ymj59ds4qtvj8tp67
zvs58fvvpc22n",
"revocationauthority": "i5vt1Cx7oDMjfFSNSFpdMFQLoyD9gqDN55",
"recoveryauthority":
"iLbcafWjFgvrZ9yGNKofWmBw1DgJyAaVe2"}}'
```

◦ The data shown in red is copied from the response message from the commit command.
◦ The data shown in green is your public address that your ID is bound to.
◦ The data shown in purple, is your private address, which can be used for private transfers, messaging, voting, polling etcetera.
◦ The data shown in Dark yellow, are the i-addresses for the revocation and recovery addresses. You can use the ID-name too,
instead of the i-addresses.

**) Some notes on IDs

You see that you can enter three IDs, viz. the main ID, the revocation ID, and the recovery ID. Actually they are equal for their own purpose,
each of them is an own, normal, independent ID, and can do anything the main ID can do on their own. They don't even have to be your IDs.
But they just can be assigned to different roles (a.k.a. authority). If you choose to do so, here's what can be controlled by them:
l The main ID is the one you would do all transactions with, and other stuff that is still to come. It always has the power to change both of
the other IDs.
l The Revocation ID can revoke the main ID, but also has control over changing anything that is related to the Revocation ID itself. This
means, if you change the Revocation ID to an ID you do not control, you no longer can change the Revocation ID again, except from the
wallet that also controls the Revocation ID. This also applies to Recovery ID.
l The Recovery ID can of course recover the main ID when it was revoked. And (only) in this process, it can also change the public and
private addresses of the main ID, and it's Revocation ID.
l Both the Revocation ID and the Recovery ID can be used for more than one main ID, so you only need to create them once. And they can
be the same, too, so you can use one other ID both for revocation and recovery.
l If you want to be able to revoke and recover your main ID, you need at least one more ID

for that. The main ID can not revoke itself.
However, if you want to have a totally self-sovereign ID, and control the private and public address it is assigned to in another way, you
can use self-assignment.
1 Revocation and Recovery ID(s) should be created before your main ID. Although there is a way to change them later, it’s rather
complicated (at time of writing this guide). So if you plan to be able to revoke and recover, it’s much easier to create them first.
Now you see what these IDs can control. What they do not control, nor revoke or recover, are the underlaying public (R-) and private (z-)
addresses that you enter on creation. These will always be in control of the wallet that has the private keys (a.k.a. WIFs) for them. They will
stay unchanged, can’t neither be revoked nor recovered. That is only possible for IDs.
You will, however, find a new address after creating an ID, which is an i-address. This address is tied to an ID, and only exists together with a
valid ID. It does not have a private key, and can not be restored other than by the associated ID.
For the experienced user it may even be an option to create an ID (e.g., for revocation or recovery) on a new wallet, with a separate
wallet.dat file. The wallet.dat can be saved and secured offline, so it is at hand when needed. But you only should do so if you’re really sure
what you are doing. If you would like to have some support with this, feel free to ask in our Discord https://discord.gg/VRKMP2S.
Now, after you know a bit more about IDs, you can decide how many and which IDs you want to create.
Finally, a few notes on referrals.
When you enter a referral ID, purchasing an ID will cost 80 instead of 100 VRSC (not including transaction fees). At the same time, the referral
ID will get 20 VRSC.
Furthermore, when someone is using your ID as a referral, you will get 20, and the referral you were using will again get 20 VRSC (but
purchase will not get cheaper than 80 VRSC).
This referral propagation will work up to the 3rd generation, i.e., up to the referrer of the referrer of the referrer. Each of them will have 20
VRSC. But at least 20 VRSC will always go back to the miners and stakers of Verus Coin, and will be added to one of the coming blocks. That’s
also why mining and staking will become more profitable with each ID that was created.
Thanks for reading, and don’t hesitate to ask in our discord if you need support! :)
veruscoin.io

Question: How do I know when an immature block will "unlock" (mature)?

Procedure:

1. Get the values of the current block height ("blocks") and the "blockstomaturity" the distance to maturity of the transaction in question. (i.e. the number of blocks between current block height and where it unlocks. You can see the number is decreasing all the time, and if you add it to the current block height you'll see a constant.)
2. Apply this formula: blockstomaturity / 1440 = days to maturity (unlocking)
   e.g. 925558/1440=643 days

Im OinkBot, An AI bot that helps everyone with everything around the veruscoin community.

Put simply, Verus is much more than any single ordinary blockchain; more of an entire ecosystem of interconnected blockchains that all operate decentralized and at the protocol level. Verus introduces Verus PBaaS (Public Blockchains as a Service), a true publicly notarized blockchain as a service with an easy to use built in wallet UI, designed to make it so that now anyone can start their very own full-fledged cryptocurrency so long as it can be funded or supported by miners. Newly created chains are just as secure as Verus itself and have the ability to be merge mined (up to 22x at once!) Benefiting both the user and the miners, self-strengthening the ecosystem and ensuring minimal fees (no room for middlemen to take a cut). Reserve backed currencies are also possible with fractional reserve capabilities and since interoperability is paramount, exchanging currencies in cross-chain transactions is as simple as sending coins from one wallet to a different coin’s wallet. Conversions are handled by a built in market maker that automatically determines price based on a predetermined curve and issues orders fairly to all buy/sells within a block, with zero spread and the added benefit of eliminating the well- established problem of front-running, just some amount of slippage based on the net buy/sell.. With reserve currencies, buyers can make buys without even needing sellers (and vice versa). Amazingly, everything is done at the protocol level on a decentralized network, meaning Verus and it's associated chains can't be censored or stopped.

VerusPay is Blockchain-integrated payment gateway for accepting Verus Coin (VRSC) in a WooCommerce ecommerce store. This plugin extends WooCommerce on Wordpress, adding the ability to accept cryptocurrency payments in Verus Coin (VRSC) using either an on-store wallet daemon (best for VPS or dedicated hosting stores) or manually configured VRSC addresses (best for shared hosting stores). When an order is submitted via the VerusPay gateway, the order will be placed 'on-hold' while awaiting payment from the customer. The customer has a limited time wherein to send the payment and the store monitors the wallet/address to confirm payment received before releasing the order and redirecting the customer to the Thank You page. VerusPay uses limited API functionality for Manual Mode, to communicate with the blockchain explorer in verifying payments and with the veruspay.io API to get up-to-date price data. These API's do not receive any private data either about the store owner, store, or customer. The only data sent to the block explorer API is the public/transparent blockchain transaction and address used. For VerusPay.io API price data, only the store-set currency is sent to retrieve the current fiat exchange rate for Verus Coin.

VerusIDs are true self-sovereign identities and aren't simply an ID system as much as a fully functional blockchain protocol. There is no business in the protocol, but plenty of opportunity for those who use what it can do for identity applications. Verus ID provides: Quantum ready friendly crypto addresses on the worldwide Verus network -- VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are

changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID -> destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs. **Fully decentralized** -- anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft. **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority. **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

A Built-in Decentralized Referral Program, Enabling Natural Growth. Verus IDs will cost Verus to acquire, 100 Verus per ID to be exact, which can be discounted to 80 Verus with referral of an existing Verus ID. The interesting twist is that all of the cost of an ID goes back into the network, either as referral fees, which are a way to get discounts or possibly even make money through referrals and built into the identity transactions themselves, or as mining and staking fees. No one besides people participating in the network as miners, stakers, or ID referrers take any proceeds from the cost of an identity. If you refer someone new and they purchase an ID with your ID as a referral, they will receive a 20% discount on the ID. In addition:\n You will receive 20 Verus directly\n The person who referred you, if there is one, will receive 20 Verus\nThe person who referred person b, if there is one, will receive 20 Verus, and\nThe miner or staker will receive the remainder of the discounted 80 Verus cost.\n As you might expect from looking at the fact that anywhere from 20 to 100 Verus goes to miners and stakers of Verus for each new identity once identities hit mainnet, some Verus blocks may have VERY high rewards for some time. The best thing about that is that regardless of how much the Verus blockchain rewards miners over and above the pre-determined coinbase reward, it will be as a result of the on-chain economy, paid for by people buying identities with no inflation of the money supply! If you or your friends missed the early days of the

Verus launch, you don't want to miss this new opportunity to be mining, staking or referring now Verus IDs are activated on the mainnet!.

To get started with veruscoin follow these links, Website: https://verus.io\n Wallet: https://verus.io/wallet\n GitHub: https://github.com/veruscoin\n Explorer: https://explorer.verus.io\n Discord: https://verus.io/discord Check https://youtu.be/YVOfIMjRf30 if you only see one empty channel.\n Bitcointalk: https://bitcointalk.org/index.php?topic=4070404.0

Verus, latin for "True", combines all of the features we believe are important in a cryptocurrency, and provides a foundation for future development. Verus is a fork of Zcash and Komodo that leverages the Komodo platform, and we appreciate the contributions and support from those teams as well as the Bitcoin developers that created a foundation for us to launch new capabilities in a system that supports Bitcoin and Zcash compatible transaction types as well as Komodo cross chain swaps and dPoW security enhancements.

If you've been around long enough and feel a little skeptical or disillusioned from a few bad experiences elsewhere, it's totally understandable. We've all had them and we know quite well that when unscrupulous projects abuse words like decentralization, interoperability, or protocol level solutions, it can have a "boy who cried wolf" effect. We believe that actions speak louder than words. Anything that may be considered hype can be backed up with functional examples. What we're doing here is genuine. We are trying to be the real deal, just as Bitcoin started out, as an open-source, fair launched, no ICO, or pre-mined, or even dev funded project. Despite this, a foundation has been established by and for fellow community members. There is always more to be done and the foundation regularly pays out bounties to community members that help to make the Verus vision a reality. In fact, just about everything has been designed to reinforce the community. Designed with efficiency in mind, only the miners and stakers are rewarded for securing the chains. This ensures bare minimum costs to the end user starting up a chain and the sleek UI removes the technical barrier.

Focus on Verus' social media presence has picked up with new, fresh looking Facebook and YouTube channels so check them out and feel free to post something or simply explore the content. There's so much going on, plenty to learn about. So much so that there is an open call to anyone and everyone out there who can submit graphical or textual content that can be used to help support the upcoming mainnet launch. A stronger effort from our already wide base of members to post and utilize social media tools to help spread the word and generate new interested users is something we can all do right now. There is no better time than now and getting fresh new minds in here to discover what is being done is the first step in getting the great snowball rolling. To those of us who truly care and believe in this project, then this message is meant for you.

Sure! Check out the brand new community website, community social media pages, new Veruscoin YouTube and Facebook pages for helpful how to or to

just keep up to date with things. Speaking of how to, we can always use more helpful videos if any community members would like to take it upon themselves to chip in. The Medium page also has lots of helpful guides and there's always something to Tweet about. Be sure to drop by the discord to meet the vibrant and helpful community in real- time, where it isn't uncommon to find the developers offering assistance. Where else can you find that? We are all working together to realize something great. If you'd like to join us and be a part of that special something but don't know how to, it can be as simple as checking out the links below and helping other's to understand that there's never been anything in existence quite like Verus, a complete, easy to use, fully decentralized blockchain ecosystem, designed specifically to benefit all participants in one of the most low cost yet efficient in every way, end-to-end systems to date. Facebook - https://www.facebook.com/VerusCoin/ YouTube - https://www.youtube.com/channel/UC_-KCHBxaDwSgNMdE3LMThg Discord - https://verus.io/discord Twitter -https://twitter.com/veruscoin Medium - https://medium.com/@veruscoin Reddit -https://reddit.com/r/veruscoin Community twitter - https://twitter.com/VerusCommunity

As a community project of just regular Joes, it can't be stressed enough how important each and every one of us all are. The true power of decentralization is putting power back in the hands of YOU.

There's a lof of information about the Veruscoin project. We have gathered as much information and resources and bundled them into this FAQ, as to give a quick overview. VRSC Wallet & data location on different OS. Linux GUI: ~/.komodo/VRSC, Mac OS: /Users//Library/Application Support/komodo/VRSC, Windows: %AppData%\Komodo\VRSC\

Windows Verus binaries: resources\app\assets\bin\win64\verusd\ contains verusd and verus Windows Komodo (and asset chains) binaries: resources\app\assets\bin\win64\ contains komodod and komodo-cli Linux Verus binaries: resources\app\assets\bin\linux64\verusd contains verusd and verus Linux Komodo (and asset chains) binaries: resources\app\assets\bin\linux64\ contains komodod and komodo-cli Note: All locations are relative to the installation location of Agama.

Windows Verus binaries: \resources\app\assets\bin\win64\verusd\ contains verusd and verus Windows Komodo (and asset chains) binaries: \resources\app\assets\bin\win64\komodod contains komodod and komodo-cli Windows Zcash binaries: \resources\app\assets\bin\win64\zcash contains zcashd and zcash-cli Linux Verus binaries: \resources\app\assets\bin\linux64\verusd contains verusd and verus Linux Komodo (and asset chains) binaries: \resources\app\assets\bin\linux64\ contains komodod and komodo-cli

Mining pool: https://luckpool.net/verus Mining pool: https://pool.verus.io/ Mining pool: https://zergpool.com/ Mining pool: https://vrsc.mcmpool.eu/ Mining pool: https://wattpool.net/ Mining pool: https://www.nlpool.nl/ Mining pool: https://vrsc.dev-codex.com/ Mining pool: http://vrsc.52hash.com/ Mining pool: https://vrsc.ciscotech.dk/ Mining pool: http://www.lepool.com.cn:8088/ Mining pool:

http://verus.bcmonster.com/ Most of the mining pools seem dead so i prefer to use the official verus community pool or luckpool.

Exchange: https://atomicdex.io/ (VRSC/any listed coin) Exchange: https://app.stex.com/de/trade/pair/BTC/VRSC/1D (VRSC/BTC) Exchange: https://safe.trade/trading/vrscbtc (VRSC/BTC, VRSC/SAFE) Exchange: https://www.aacoin.com/#/trade?symbol=VRSC_BTC (VRSC/BTC) Exchange: https://graviex.net/markets/vrscbtc (VRSC/BTC, VRSC/KMD, VRSC/USD) Exchange: https://www.kuangex.com/#/exchange/vrsc_usdt (VRSC/USDT)

Komodo is a decentralized public blockchain project that is community-driven and utilizes the VerusHash 2.1 algorithm for mining. VerusCoin, the native currency of the Verus blockchain, has a fair launch with no ICO and no premine. VerusCoin's privacy features are enhanced with Sapling and zk-SNARKs zero-knowledge proofs. The block time is 1 minute, and the block reward is 24 VRSC, with a total supply of 83,540,184. The reward emission schedule started with a linear ramp and has since halved every month for the first five months and every two years after that. All rewards at or above 192 are time-locked for random release between block 129,600 and 1,181,520. The Verus blockchain's unique feature is VerusID, which is a fully functional blockchain protocol that provides numerous opportunities for identity applications. VerusCoin's website is https://verus.io/, and the project's code is available on Github at https://github.com/veruscoin. The project has three block explorers, https://explorer.verus.io/, https://explorer.vrsc.0x03.services/, and https://dex.explorer.dexstats.info/. VerusCoin's community can be found on Discord at https://discord.gg/VRKMP2S, on Twitter at https://twitter.com/veruscoin, on Medium at https://medium.com/@veruscoin, and on Reddit at https://reddit.com/r/veruscoin.

VerusCoin can be mined with CPUs, GPUs and FPGAs, solo and in pools. However, the algo is carefully designed for CPUs, and they still substantially outrange GPUs. ARM mining works in general, but although fairly efficient, without high hashrates. FPGAs can mine this coin, but don't outperform CPUs by much.

Sure! Besides solo-mining with Verus Desktop or CLI wallet, you can use CCminer for CPU (efficient for most modern CPUs) or GPU that can be found on https://github.com/monkins1010/ccminer/releases or you could use NHeq miner that can be found on https://github.com/VerusCoin/nheqminer/releases (for Windows, Linux, and MacOS). A spreadsheet to compare hashrates can be found here: https://docs.google.com/spreadsheets/d/1RrSYJDV0Mjj3X-myMC3aQDGkcipivxHsD7ZxJ3r5f_A/edit#gid=201266774 You can also compare older and current algos.

Certainly! Verus Desktop GUI wallet is a newly developed multi-coin wallet supporting VerusID https://github.com/VerusCoin/Verus-Desktop/releases (for Win, Linux, Mac and ARM Linux) the CLI wallets can be found here: https://github.com/VerusCoin/VerusCoin/releases/ There's also possibilities to test coming PBaaS functionality in this wallet.

A beta version of the Android mobile wallet can be found here: https://github.com/VerusCoin/Verus-Mobile/releases A beta version of the iOS mobile wallet can be accessed via this Apple TestFlight invite https://testflight.apple.com/join/ZS43lYcw The Wallets will come to the playstore and the appstore soon.

Our Paper Wallet can be accessed here: https://paperwallet.verus.io

If you need a bootstrap, you can find it here: https://bootstrap.verus.io (a guide how to apply is pinned in #community-support channel in Discord: https://discord.gg/VRKMP2S or in the HOW-TO & FAQ section of our website https://wiki.verus.io/how-to/how-to_bootstrap.md)

Yes! For those interested in VerusPay, a guide can be found here: https://veruspay.io/setup/

And finally for those interested in running various Veruscoin services: https://github.com/VerusCoin/VerusServicesSetup

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides: Quantum-ready friendly crypto-addresses on the worldwide Verus network and Fully Decentralized Protocol

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are: Revocable -- each ID includes a revocation authority, which defaults to the identity self. If another ID is specified as the revocation authority it can be used to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft or turnover in an organization. Recoverable -- each ID also includes a separate recovery authority, which also defaults to self. If another ID is specified as the recovery authority it can be used to recover the ID from its revoked state, with the option to alter the

primary authorities used to spend and sign. Private - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong identity, attested to with a quantum secure signature when that is available. Powerful - Multiple addresses or other IDs can be defined as primary addresses, and any number of those may be required to spend, sign, or alter the identity (N of M). The revocation authority may only be altered by the revocation authority, and the same applies to the recovery authority, either of which may be another identity with its own N of M multisig controls for its primary addresses.

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft. Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity. Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built. For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes. In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret,

the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus-Desktop Procedure: In Verus-Desktop, exit your profile, by selecting the exit icon at ther top right. Wait a minute or two, allowing the wallet to close completely in the background. click help, Bootstrap VRSC. That opens up a new window. Follow the instructions and when finished successfully, select your preferred profile and enter Verus Desktop. Verus CLI Procedure: Go to the folder where your daemon is extracted (standard verus-cli) Shutdown verusd and wait for it to close completely doubleclick fetch-bootstrap in your file browser. Follow the instructions and when finished, start your verusd daemon as usual.

Make sure your wallet is not active. If you already had you wallet running, backup essential files: a. Go to VRSC Wallet location b. copy wallet.dat to a SAFE location c. copy VRSC.conf to a SAFE location d. Verify that both files are copied to your safe location. Make sure the latest version of your Wallet for Verus is installed a. Download the latest Verus Wallet from link 1, supplied above. b. Verify the SHA256 checksum & signature of your download, to verify you have an untampered installer. c. extract the file you just downloaded to a suitable location. On MacOS and Linux you will have extracted an AppImage which can be run directly. Windows users need to run the installer. Installing the bootstrap: a. Download the bootstrap from Link 2. (For Windows a zip archive is available to accomodate native extraction. Linux and MacOS users can use the tar.gz archive) b. (Optional, but recommended) Verify the md5, sha256 or sha512 checksum and the signature of your download, to verify that you downloaded an untampered Bootstrap archive. c. Remove all files and folders from VRSC Wallet Location except wallet.dat, debug.log, VRSC.conf and if applicable VRSC-bootstrap.*. d. Extract the downloaded archive to VRSC Wallet location. Make absolutely sure the folders blocks and chainstate are extracted into the correct folder. If the end up in a different folder (eg. VRSC-bootstrap-folder) move them to VRSC Wallet location. f you had a VRSC wallet running before, restore essential files: a. Go to VRSC Wallet location b. Verify that your wallet.dat is bigger than the one in this folder (if any is present) c. copy wallet.dat from your SAFE location. Now Start your wallet.

Using Verus-Desktop: download the new version, extract the archive, verify the signature using the data in the *.signature.txt-file through your existing Verus-Desktop, VerusID tab, Verify Signed Data and choose to verify a file. Only continue when this verification returns True. Stop your Verus-Desktop Wallet, Install the verified installer on windows. Start your wallet. Using CLI Wallet: download the new version, extract the archive, verify the signature using the data in the *.signature.txt-

file with the command ./verus verifyfile "address or identity" "signature" "filepath/filename" command. Only continue when this verification returns True. stop your verusdaemon verus stop, extract the verified archive to your current CLI-wallet location. Start your wallet (verusd).

In Verus Desktop: Go to Settings, Coin Settings and click Export native wallet backup. Confirm that you want to export after reading the pop-up. The green message will tell you where the backup is and what it's name is. In linux or MacOS CLI: run ./verus z_exportwallet '<mywalletexport>' and in Windows CLI you just run verus z_exportwallet '<mywalletexport>'

There are 2 ways to do it, first way is to use your backup of your wallet.dat file. Stop verusd. For Windows-Desktop or Agama, just exit and wait for it to close completely. For the linux cli run ./verus stop, or for the windows cli run verus stop. Once your wallet is finished closing copy the backup of your wallet.dat file from your backup location to the directory listed in the start of this document (see above). Now restart your wallet by launching Verus Desktop, Agama or running verusd for the CLI. Second way is to use your walletexport file, Note: The filename you replace<mywalletimport> with, can only contain letters and figures, no other characters, so it cannot have an file-extension Attention: The command z_importwallet triggers the wallet to rescan in order to make all transactions to the freshly imported wallet addresses visible. Rescanning your wallet may take a considerable time, during which your wallet may not respond to other commands. Please be patient. The <PATH> in the z_importwallet command needs to be the full absolute path to the file. replace LOGINNAME with the actual loginname. In Verus Desktop: Go to Settings, Coin Settings and click Import native wallet backup. Click Choose file, browse to your backup file, select it and click Open. Click Import to start the inport process. In Linux or MacOS CLI run ./verus z_importwallet '<PATH><mywalletimport>'. Or in Windows CLI run verus z_importwallet \'<PATH><mywalletimport>\'

Make sure you have your seed phrase and password you use to log into your Lite mode wallet available. First of all make a notition of your address and balance of VRSC you have in your wallet, before closing Verus Desktop. Make sure the latest version of Verus-Desktop is installed. Download the latest Verus-Desktop. Verify the signature of your download, so you have an untampered installer. https://verus.io/verify-signatures. Run the file you just downloaded to install it. Installing the bootstrap: Download the bootstrap from https://bootstrap.verus.io Verify the SHA256 checksum of your download, to verify you have an untampered Bootstrap. Unpack the downloaded archive to VRSC wallet & data location. Getting Verus-Desktop ready for Native mode: Start Verus-Desktop and enter your profile (if not loaded automatically). If present in your profile, deactivate Verus Lite. Click + Add Coin, select Verus from the dropdown list and continue. Select Native and optionallu tick the options Start staking, Start Mining and fill in the amount of threads to mine with. Click Add Coin. Verus-Desktop will add Verus as Native chain to your screens. You may get a red warning message about Zcash params. (Verus Desktop will detect if you have the necessary ZCash parameter files and download them if needed) As soon as the download is finished, Verus-Desktop will continu and bring you into your wallet. It will

automatically start to synchronize the blockchain. Since we already put the majority of the chain in place, this will take just a few minutes. Now you just want to import the lite wallet address in verus desktop native. Just ask me that and i will let you know!

Important General Information

verus command "<userinput>" needs to be entered literally, with <userinput> replaced by your specific userdata. So if the text directs you to use for example "<Public Address>", you replace that (including the < and >) with the address,
so it looks similar to this: "RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r".
This method is confirmed to work on Verus and all Komodo assetchains.

Prerequisites

In order to convert your seedphrase into a Private key (WIF), you need to have a running native wallet first, that is fully synchronized. The earliest wallet that supports these functions is Verus Desktop v0.6.4-beta-1.
If needed, use this guide to quickly synchronize your wallet:
https://wiki.verus.io/#!how-to/how-to_bootstrap.md

Converting Seed to WIF

If you have a seed, you can retrieve your Private key (WIF) by having the Verus Desktop wallet convert it for you.
To convert your seed phrase in Verus Desktop, go to settings --> Coin Settings, select the chain you want to use in the top-right corner and enter the following command:
run convertpassphrase "word_1 word_2 word_3 ... word_n"
Note: Make sure you replace Word1 word2 word3 ... word_n with the actual seedphrase (12 or 24 words) of the address you want to import!
You will receive a response similar to this:

```
{
"walletpassphrase": "seedphrase",
"address": "RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r",
"pubkey":
"02ffc2f4b071afdec631e3fb7d435a0047be14a81ea1a269e4206b0068c0c1fa6f",
"privkey":
"d899ed88e9ee2e90c2cf51cb47e7b4495ec1e1cb10763bb1c111b0bde48bf86c",
"wif": "UwGb5KvGPfMUr1tu74Desjh87ZeJM4wq5goLyThcogeLifc5aJqT"
}
```

Copy that information and store it somewhere SAFE. With this information anyone having access to it will have full control over that address.
The 52-character string after "wif": that is shown, is what you want to import in the next step.

Importing a single WIF for a public (transparent) address

To import your address, go to settings --> Coin Settings, select the chain you want to use in the top-right corner and enter the following command:
run importprivkey "<wif>" "" true
Replace <wif> with the actual wif you got from the convertpassphrase command earlier.
Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.
Note: The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite

some time for the process to finish in the background, especially if the address has many transaction on it.

Importing multiple WIFs in one batch for public (transparent) addresses

To import your address, go to settings --> Coin Settings, select the chain you want to use in the top-right corner and enter the following command for every WIF except for the final one:

run importprivkey "<wif>" "" false

Import the final WIF with this command:

run importprivkey "<wif>" "" true

The last command triggers the chain to rescan all addresses in your wallet, including all the addresses you just imported.

Replace <wif> with the actual wif (like the one you got from the convertpassphrase command earlier).

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Importing a single WIF for a private address

To import your address, go to settings --> Coin Settings, select the chain you want to use in the top-right corner and enter the following command:

run z_importkey "<wif>" "yes" 1

Replace <wif> with the actual wif you got from the convertpassphrase command earlier.

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Importing multiple WIFs in one batch for private addresses

To import your address, go to settings --> Coin Settings, select the chain you want to use in the top-right corner and enter the following command for every WIF except for the final one:

run z_importkey "<wif>" "no"

Import the final WIF with this command:

run z_importkey "<wif>" "yes" 1

The last command triggers the chain to rescan all addresses in your wallet, including all the addresses you just imported.

Replace <wif> with the actual wif (like the one you got from the convertpassphrase command earlier).

Note: Don't use the WIF from the example above, but use the one from the CLI-interface in Verus Desktop.

The GUI wallet will not show any progress on the import and may give messages that the RPC daemon is not reacting. It will take quite some time for the process to finish in the background, especially if the address has many transaction on it.

Notice: Read it completely before use.

Important General Information

wallet.dat location on Linux: ~/.komodo/VRSC

Procedure:

First make sure your system is up to date:
sudo apt-get update && apt-get upgrade -y
I suggest not to use the root account. I you have not yet done, set up a new user
sudo adduser newusername
sudo usermod -aG sudo newusername
Switch to new username
su - newusername
Test the your new user actually has root (SuDo) access, e.g.:
sudo ls -la /root
You should get some lines like this:
drwx------ 6 root root 4096 Jul 3 15:56
Download & install the wallet binaries:
wget https://github.com/VerusCoin/VerusCoin/releases/download/v0.9.3/Verus-CLI-Linux-v0.9.3-amd64.tgz
The downloaded archive contains another archive and a signature text file, enabling the archive within to be verified (You'll need a running wallet to do that)
Also: Verify the URL to the latest version from the Download latest Wallet above.
tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tgz
Now extract the wallet archive:
tar -xvf Verus-CLI-Linux-v0.9.3-amd64.tar.gz
Change directory to verus-cli
cd verus-cli
Fetch parameters, takes time, more on slow Internet connection
./fetch-params
Creating the chaindata directory
cd ~
mkdir -p .komodo/VRSC
cd ~/.komodo/VRSC
Download the block-chain bootstrap, this considerably speeds up synchronisation of the block-chain from days to minutes... (optional)
wget https://bootstrap.verus.io/VRSC-bootstrap.tar.gz
tar -xvf VRSC-bootstrap.tar.gz
Install libraries for Verus
sudo apt-get install libcurl3 g++-multilib -y
Install Tmux a terminal multiplexer with which you can run threads in the background see https://en.wikipedia.org/wiki/Tmux
sudo apt-get install tmux -y
Start tmux:
tmux
Launch Verus Daemon with or without number of threads
(usually number of threads equals number of cores or double of that if the processor support hyper threading well)
~/verus-cli/verusd -gen -genproclimit
~/verus-cli/verusd -gen -genproclimit=24
Once mining is operational â€“ again this may take some time â€“ youâ€™ll see: 256 mega hashes complete - working
then detach tmux
[ctrl]&b d
Disable login with Root User
(make sure your newly created user login works and has sudo rights)

sudo nano /etc/ssh/sshd_config
Find: PermitRootLogin yes
And set to
PermitRootLogin no
Apply new settings:
sudo systemctl restart sshd

Note: Shielding is no longer required for coinbase rewards after block 800200. Earlier timelocked coins will still need to be shielded in order to use them.

Important General Information
verus command "<userinput>" needs to be entered literally, with <userinput> replaced by your specific userdata. So if the text directs you to use for example "<Public Address>", you replace that (including the < and >) with the address,
so it looks similar to this: "RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r".

General remarks on CLI wallet:
On Windows command line enter the commands as shown without the surrounding quotation marks
In Linux shell preceed the commands without surrounding quotation marks with ./
In MacOS shell preceed the commands without surrounding quotation marks with ./
Example: the windows version verus listtransactions transforms in Linux or MacOS to ./verus listtransactions.

General remarks on Windows command line formatting:
The CLI help shows the command format for Linux and MacOS.
For windows substitute the shown '-character with the "-character.
For windows substitute the shown "-character with the \"-characters.

Procedure:
You must first "shield coins" (send from a transparent R-addr to a shielded zaddr) to be able to use them in staking, when they first unlock.
This is part of the Zcash protocol itself. The commands below assume you are in the Verus source code directory

verus z_shieldcoinbase

You can either use an existing zaddr or use a new zaddr. To make a new zaddr:

verus z_getnewaddress

Use the address the above command outputs in the z_shieldcoinbase command

To shield all coinbase in your wallet, you can use "*" (quotes are important) and zaddr that is getting the funds:

verus z_shieldcoinbase "*" <YOUR zs-ADDRESS>

To just shield a single address, specify that as the first argument:

verus z_shieldcoinbase <YOUR R-ADDRESS> <YOUR zs-ADDRESS>

Once the funds have moved to the zaddr and are confirmed, you can freely send them to any address, They will be eligble for staking id sent to a transpartent address (R-address).

To send from a certain zaddr to a transparent address, use z_sendmany. The following command sends 10 Verus to a given transparent address, ID address or ID-name.
Example: verus z_sendmany zcZpfuzzJqmNJ3fUJekvbnyuxuJe9eAURAHrMCvN2Nr7VuWjakb1LEw6j2etPcCnr45BRot7MaMbipuS5da162BfuUkFGxx '[{"amount":10,"address":"Verus Coin Foundation@"}]'

Important General Information
VRSCTEST data location:
Linux GUI: ~/.komodo/VRSCTEST
Mac OS: /Users//Library/Application Support/Komodo/VRSCTEST
Windows 10: %AppData%\Roaming\Komodo\VRSCTEST\

General remarks on CLI wallet:
On Windows command line enter the commands as shown without the surrounding quotation marks
In Linux shell preceed the commands without surrounding quotation marks with ./
In MacOS shell preceed the commands without surrounding quotation marks with ./
Example: the windows version verus listtransactions transforms in Linux or MacOS to ./verus listtransactions.

General remarks on Windows command line formatting:
The CLI help shows the command format for Linux and MacOS.
For windows substitute the shown '-character with the "-character.
For windows substitute the shown "-character with the \"-characters.

Necessary files:
Link 1: Download latest Wallet

Procedure:
Joining Verus testnet to test the latest capabilities before they are released to mainnet or simply test if your goals are possible without spending VRSC (testnet coins hold no value) is easy.

Download a wallet
The first thing you need is a VRSC wallet. The CLI-wallet and Verus Desktop GUI wallet are available on the link above for Windows, Linux and MacOS. If you already have a wallet verify that the wallet is the most recent version and update if needed.

Verus Desktop Wallet
Start your Verus Desktop wallet.
If you have never run Verus testnet on your system before:

Go to settings (cogwheel icon) and select General Settings.
Select Enable VRSCTEST.
Click Save Changes
Restart Verus desktop
When logged in, click Add Coin, select Verus Testnet and click Continue.
Select the startup parameters you desire (Native (Lite mode is not available on testnet), Staking, mining (specify number of threads), reindex blockchain and/or rescan wallet) and click Add Coin.
CLI wallet
1. start CLI walletdaemon using these parameters:
verusd -chain=VRSCTEST
Any extra parameter that you are used to for VRSC (like -mint or -pubkey=) can be appended as well.
2. commands through the CLI are in the following format:
verus -chain=VRSCTEST
The only difference with the normal VRSC chain is the -chain=VRSCTEST option, that is added.

Get the values of the current block height ("blocks") and the "blockstomaturity" the distance to maturity of the transaction in question. (i.e. the number of blocks between current block height and where it unlocks. You can see the number is decreasing all the time, and if you add it to the current block height you'll see a constant.)
Apply this formula: blockstomaturity / 1440 = days to maturity (unlocking)
e.g. 925558/1440=643 days

One block = 12 coins (as now)
NetworkHashrate = retrieved by getmininginfo command from </>CLI
LocalHashrate = retrieved by getmininginfo command
BlockTime = 60 seconds (average)
Real example with - 31 threads AMD Ryzen 5950x @ 4.4Ghz -
Bear in mind that these are average times to find a block. In real life you may hit a block much sooner or later after finding the last. In the long run it averages out to the values predicted.

Verus Wallet.dat, Chaindata & VRSC.conf standard locations
Linux: ~/.komodo/VRSC
Mac OS: ~/Library/Application Support/Komodo/VRSC
Windows 10: %AppData%\Roaming\Komodo\VRSC\
OS independent through Verus Desktop: Click help, Show Verus data folder (default)
Prerequisites
Have a native VRSC wallet running
In Verus Desktop
In Verus Desktop, there is at this moment no way to enter an address to mine to. However, editing the VRSC.conf can be done to achive our goal:
In the receive window of your wallet, click the hamburger (three vertical dots) next to the address you want to receive your rewards in and click Copy Public Key. Close down Verus Desktop.
Edit VRSC.conf(see standard locations at the top) and add the line pubkey=THELONGSTRINGCOPIED. Save and exit.
* Start Verus Desktop as you normally do.

In Agama Wallet
Step 1 - First get your wallet address you want to mine to:
* If you don't have an address, click "Receive", click "Get New Address" and choose "Transparent Address" from the drop down.

Step 2 - Next we need to retrieve our pubkey,
* click on the hamburg next to the address that you want to receive the rewards in and click copy pubkey

Step 3 - Set your PubKey
Go to 'Settings', 'App Config (config.json)' and enter your pubkey(THELONGSTRINGCOPIED) into the 'Pubkey VRSC mining key' field.
Click 'Save app config' to save these settings.
* Restart Agama

In Verus CLI
Step 1 - First get your wallet address you want to mine to:
You can find an address if you already have previous transactions, or you can create a new one. To find an address from a previous transaction, use the command line verus listtransactions and copy the address found after "address".
To generate a new wallet address, use the command line verus getnewaddress and a new address will be created.

Step 2 - Next, using your new address, enter the command with verus-cli verus validateaddress. From the output find the long string after "pubkey", copy without the quotation marks.

Step 3 - Set your PubKey
Option 1: use this pubkey when starting your daemon by adding the following line to the end of your command, just before the "&" sign: -pubkey=THELONGSTRINGCOPIED Option 2: edit your VRSC.conf and add the line pubkey=THELONGSTRINGCOPIED. Then start your whallet as you are used to.

Your rewards will now be mined to that address. It would be a good idea to keep notes and associate the wallet address with the pubkey...also to double check that you did validate the correct pubkey for the wallet address, making sure you made no errors.

Verus Wallet.dat, Chaindata & VRSC.conf standard locations
Linux: ~/.komodo/VRSC
Mac OS: ~/Library/Application Support/Komodo/VRSC
Windows 10: %AppData%\Roaming\Komodo\VRSC\
OS independent through Verus Desktop: Click help, Show Verus data folder (default)
Procedure
With one of the wallets loaded, issue the following command:
z_exportwallet FILENAME (ie z_exportwallet export_instance01, the filename cannot have a . in it.)
Copy the generated file to the machine that hosts your main wallet. This file will either be in the same directory as the config file and wallet.dat file, or in a location specified by exportdir in VRSC.conf.
Issue the following command (on the "main" verus-cli): z_importwallet /LOCAL_PATH/EXPORTFILENAME (ie /home/user/export_instance01)

note: Older versions of verusd required expordir to be set in VRSC.conf before exporting a wallet. If you get an error about your export directory not being set, please upgrade immediately.

These commands can be given in:
CLI wallet: ./verus z_exportwallet FILENAME& ./verus z_importwallet /LOCAL_PATH/FILENAME
Verus Desktop in settings, coin settings: run z_exportwallet FILENAME& run z_importwallet /LOCAL_PATH/FILENAME
Verus Agama in settings, <CLI>: z_exportwallet FILENAME& z_importwallet /LOCAL_PATH/FILENAME

The reward received depends of the blocknummer:

Era 1:
1st week: Block 0 - 10080 ==> 0 to 384 VRSC reward ==> 16,588,800 VRSC total this period (reward rising linearly and changing each block)

Era 2:
1st month: Block 10080 - 53279 ==> 384 VRSC reward ==> 8,294,400 VRSC total this period
2nd month: Block 53280 - 96479 ==> 192 VRSC reward ==> 4,147,200 VRSC total this period
3rd month: Block 96480 - 139679 ==> 96 VRSC reward ==> 4,147,200 VRSC total this period
4th month: Block 139680 - 182879 ==> 48 VRSC reward ==> 2,073,600 VRSC total this period
5th month: Block 182880 - 226079 ==> 24 VRSC reward ==> 1,036,800 VRSC total this period

Era 3:
Years 1+2: Block 226080 - 1277279 ==> 24 VRSC reward ==> 25,228,800 VRSC total this period
Years 3+4: Block 1277280 - 2328479 ==> 12 VRSC reward ==> 12,614,400 VRSC total this period
Years 5+6: Block 2328480 - 3379679 ==> 6 VRSC reward ==> 6,307,200 VRSC total this period
. . . halving indefinitely every 1051200 blocks (approximately 2 years)

All rewards equal or over 192 VRSC are time locked to mature at a random block between 129,600 and 1,181,520

Use with: https://veruspay.io/api/ for simple USD VRSC price, or choose options now added!

Options (values are case insensitive):
currency - BTC or Fiat code like USD or CAD
ticker - ARRR or VRSC
data - volume or price - volume only relevant if exchange is defined
exch - name of supported exchange, e.g. digitalprice - If no exchange, price is average of all supported for that coin.

If no options are set, the default is average price in USD fiat of VRSC.

Examples:

https://veruspay.io/api/ - This get's the current price of VRSC in USD, weighted against 24hr volume across all exchanges. This is the default return.
https://veruspay.io/api/?exch=digitalprice&currency=cad - This will get the current price on digital price for VRSC and display in CAD fiat
https://veruspay.io/api/?currency=btc - This will get the average price of VRSC in BTC, weighted by 24 hr volume across both exchanges
https://veruspay.io/api/?currency=cad - This gets the current average price of VRSC in CAD, weighted by 24 hr volume across both exchanges
https://veruspay.io/api/?exch=cryptobridge&data=volume - This will get the 24 volume of VRSC on CryptoBridge in the default currency of USD
https://veruspay.io/api/?exch=cryptobridge&data=volume&currency=btc - This does the same but with BTC as the currency result
https://veruspay.io/api/?currency=cad&ticker=arrr - Gets the average price of ARRR.

Mining VRSC with Verushash is at this time one of the most profitable coins to mine with your CPU.
If you want to know how profitable it is, you need to know a few important details about your own conditions.

1. CPU - If that is a fairly modern CPU with AES and AVX instructions built in.
This will be true if your processor is produced in 2013 or later. For processors between 2008 and 2012 you need to check the specifications. Older processors can still mine, but they will not perform well.
2. GPU - It is possible to mine with fairly modern Nvidia GPU's, but since Verushash uses specific functions from the
mentioned AES and AVX instruction sets, they will perform better than old CPUs, but worse than modern CPUs.
3. FPGA - Many of these semi-specialized machines can be reprogrammed to mine VRSC. They don't have the same power per processor as a CPU, but they often have multiple processors running parrallel. The stronger ones can outperform a CPU easily, but need a lot of power to do so.

If you know the answer to the hardware, you can look op a comparable one in the Spreadsheat from Link 1, to give you an
idea of the performance.

If you know your energy price also, you can calculate an estimation of how much your hardware can earn you at this moment.

If you have to get new hardware just to mine, think about it for a moment: would you have bought new hardware anyway? If so,
you can use this information to get an idea what kind of hardware you want to buy.

You can see the following characteristics:
1. a block hash without a lot of leading zeros
2. a last transaction that in your wallet has no fee, is from the same address that it is to, which is also the address the coinbase is to (easiest confirmation) and

3. The POS difficulty is encoded in the low 32 bits of the nonce with 96 of the next nonce bits as zero/reserved, then a random hash at the top half

You'll start staking with the first VRSCs that are not time locked + in your public / transparent wallet + 150 blocks old (or about 2.5 hours).

Your chances to win a block: Your coins in your public/transparent wallet / Total staking supply in public/transparent wallets (which is max. about 485.000 VRSC from the first (sunrise) week as long as the rewards were below 192)

Remember: There will be only one reward every minute. Itâ€™s going to be either mining or staking, so on average 720 mining and 720 staking rewards every day.

Example: I have 300 coins in a public/transparent address / 300.000 in public wallets (let's assume some part is lost/not staking or in private wallets), so that would be 1/1000 x 720 of a chance or around average 1,4 days for a staking block rewards. Hash power does not influence staking reward.

Regarding the Verus debug.log: â€œ No eligible staking transaction foundâ€œ. It means that you are staking but have not received a reward yet. @miketout will change the message soon.

Regarding time locked coins:
The Zcash protocol requires you to send all coins received by mining (on wallet, not pool mining) or staking (reward transactions, also on wallet, not pool staking) once unlocked to a private address and then to a public/transparent address before you can use them either for staking or for making transactions (thatâ€™s how you make use of your rewarded coins = coinbase coins). So, once your coins loose their time lock, you can unlock those coins as described in ["Shield Verus Coins via Command Line Interface]"(#!how-to/how-to_shield_via_cli.md). Once you've transferred the coins from your private address back to (one of) your public / transparent address(es) and youâ€™ll automatically start staking.

CLI:
You can check this in the VerusExplorer https://explorer.verus.io/
1. Check last 10 transactions in the </>CLI. Use listtransactions.
2. Copy the blockhash of the received award

Hint: An example:
https://explorer.verus.io/api/getblock?hash=9e6fa91356211a554c580c90ec9c2067dd420ff74c7d33481775793f7b0e7f03 â€" so this one is minted....

Verus Desktop:
In Verus Desktop, simply go to your Mining Dashboard and enter into Verus details.
Scroll down to the bottom of the page. It will list the rewards as mined or minted in green.
The TXIDs that staked the minted rewards are shown in blue.

Verus Agama (Deprecated):
In the GUI you can also click yourself thru to that information.
1. Click on the magnifying glass all the way on the right of the transaction.
2. On the pop-up click on "Open in the VRSC Explorer" bottom left.
3. In the VRSC explorer click on the block hash value (in light blue) â€" now the block hash is displayed as the title of the box.
4. Click on the info "i" on the right and click on "search": in the result displayed for blocktype 'mined' or 'minted'.(edited)

Verus Desktop
Verus Desktop shows your immature balance in your wallet dashboard, between the Transparant Balance and Private Balance.

Agama (Deprecated)
In the Agama wallet click on the hamburger menu (the three stripes) on the top right
click on Settings
click on the item </> CLI
Select VRSC as coin
Type the following command: getwalletinfo
click Execute
scroll down and find "immature_balance": which will give you the amount of time-locked coins in your wallet.

Important General Information
verus command "<userinput>" needs to be entered literally, with <userinput> replaced by your specific userdata. So if the text directs you to use for example "<Public Address>", you replace that (including the < and >) with the address,
so it looks similar to this: "RYX6RYU3AAvwVCNyNM4cVyGUhSMUPvKs3r".

Remarks on Windows command line formatting:
The CLI help shows the command format for Linux and MacOS. On the native windows command prompt (cmd.com) the formatting is different.
In windows command prompt, substitute the shown '-character with the "-character. In windows command prompt, substitute the shown "-character with the \"-characters.
* In windows command prompt, omit the preceding ./.

Note: As an example, in Linux the command:
./verus z_sendmany <my_private_address_without_quotationmarks>
'[{"address":"<my_transparent_address>","amount":<95.9998>}]'
should be entered on the Windows command prompt as:
verus z_sendmany <my_private_address_without_quotationmarks>
"[{\"address\":\"<my_transparent_address>\",\"amount\":<95.9998>}]"

Procedure:
1) ./verus z_shieldcoinbase "*" my_private_address_without_quotationmarks
this capture all so called coinbases, i.e. mined coins that are not yet staking.
you have to wait 100 blocks (minutes) after receiving them before being able to move them.
wait for a few minutes for the tx to be confirmed.

2) ./verus z_getbalance <my_private_address_without_quotationmarks> this is to substract the 0.0001 VRSC fee from the balance in the next step.
3) ./verus z_sendmany <my_private_address_without_quotationmarks> '[{"address":"<my_transparent_address>","amount":<95.9998>}]'
amount is minus 0.0001 from balance and without quotation marks.
(4) to verify: ./verus z_gettotalbalance
after a few minutes (operations from private addresses are a bit time consuming)

This solution can be solved in 2 ways: you can simply install the latest bootstrap file (less work, big download) or search manually for the forked block and invalidate that block (time-consuming, no download needed.)

Procedure 1 (Easy, installing bootstrap)
For all GUI or CLI users.
1. Stop the wallet/mining process by cleanly shutting down the program.
2. Update your wallet if necessary.
3. Follow the procedure in HOW-TO Backup, Install or Update and Bootstrap your wallet.md to efficiently rectify the problem.
4. Do not be dismayed if it seems that your mining rewards suddenly seem to come to a halt. Remember, when you mine to the wrong chain rewards can come in very quickly, but they are worth nothing.

Procedure 2 (Time consuming, no extra download)
Verus-Desktop
The commands are all entered in the Native Client Terminal that is located under Settings, Coin Settings.
Search for the earliest block that not matches the blockchain:
run getblockhash <suspected blocknumber> will show you the blockhash for the blocknumber you filled in
The response shown in the Native Client Terminal will be similar to this:
5cc7844973fb95ef17f1772ea4aba579f0d8273fb0ee6064cd8e707d1056c646
Check the blockhash your command gave you against the blockhash the explorer shows.
If the blockhash from the explorer is different than yours, repeat steps 2 & 3 until you find the earliest block that is different.
Use the earliest incorrect blockhash from your system to invalidate that block:
run invalidateblock <earliest incorrect blockhash>
The Native Client Terminal will not give feedback on this command.
Now use the correct blockhash that the explorer gave you for the block you just locally invalidated:
run reconsiderblock <correct blockhash>
Again the Native Client Terminal will not give feedback on this command.
Once your wallet connects to a node that is on the correct chain, it will quicly synchronize.
If needed you can either restart your wallet to force new connections or manually disconnect bad nodes.
CLI
Search for the earliest block that not matches the blockchain:
./verus getblockhash <suspected blocknumber> will show you the blockhash for the blocknumber you filled in
The response will be similar to this:

5cc7844973fb95ef17f1772ea4aba579f0d8273fb0ee6064cd8e707d1056c646
Check the blockhash your command gave you against the blockhash the explorer shows.
If the blockhash from the explorer is different than yours, repeat steps 1 & 2 until you find the earliest block that is different.
Use the earliest incorrect blockhash from your system to invalidate that block:
./verus invalidateblock <earliest incorrect blockhash>
The deamon will not give feedback on this command.
Now use the correct blockhash that the explorer gave you for the block you just locally invalidated:
./verus reconsiderblock <correct blockhash>
Again the daemon will not give feedback on this command.
Once your daemon connects to a node that is on the correct chain, it will quicly synchronize.
If needed you can either restart your daemon to force new connections or manually disconnect bad nodes.

luckpool.net
pool.verus.io
zergpool.com
wattpool.net
vrsc.ciscotech.dk
www.lepool.com.cn
www.zhuaao.com
aod-tech.com
verus.alphatechit.co.uk
MadCatMining (currently inactive)

Dudezmobi
Ginasis
A staking pool has advantages and disadvantages over solo staking:

Disadvantages:
1. It causes centralization on the network
2. The pool owner is in control of all funds in the pool, including yours.
3. You need to trust the pool operator to share the rewards fairly.
4. You need to trust the pool operator to release your funds back to you on request.
5. You need to trust the pool operator to securely run the pool 24/7.

Advantages:
1. You don't need to run your wallet 24/7 in native mode.

The CLI help shows the command format for Linux and MacOS. On the native windows command prompt (cmd.com) the formatting is different.
For windows substitute the shown '-character with the '-character.
For windows substitute the shown '-character with the \'-characters.

In order to let your mining not interfere with other processes running on your PC, we'll need to deprioritize
the mining process. This will result in your miner throttling down whenever your PC needs processing power.

People have reported to be able to use CPU-heavy applications, like games, without the miner interfering, with
this method.
Because the Verus wallet (GUI and CLI) does a lot more than mining, it is not recommended to use this for a
solo-mining setup. This is tested on Verus NHEQminer and Verus CCMiner on Windows .
First thing you will need is pool-mining software. The link is supplied above this procedure. Download your
preferred miner, extract it to your prefered location and configure the batchfile with the pool of your choice
and your own mining address.
Now we'll need to adjust the batchfile to run on low priority:
NHEQMINER:
change to the folder you extracted the download
edit the start.bat
Scroll down to the line, that needs to be adjusted:
nheqminer.exe -v -l %PoolHost%:%Port% -u
%PublicVerusCoinAddress%.%WorkerName% -t %Threads% % % % % % % % % %
adjust the line so it looks like this:
%windir%\system\cmd.exe /c start 'NHEQminer VRSC' /Low
'%THIS_DIR%\nheqminer.exe' -v -l %PoolHost%:%Port% -u
%PublicVerusCoinAddress%.%WorkerName% -t %Threads% % % % % % % % % %
save and exit the file
run start.bat to start mining
CCMINER:
Change to the folder you put the three files from the downloaded archive in (run.verushhash.bat, libcrypto-_-x.dll & ccminer.exe)
edit the run.verushhash.bat
The line, that needs to be adjusted:
ccminer -a verus -o stratum+tcp://na.luckpool.net: -u
REoPcdGXthLyeTCrJtrQvxhYTknbFbec.Donator -p d= -t
adjust the line, so it looks like:
C:\Windows\System\cmd.exe /c start 'ccminerCPUv. VerusPool' /Low
'c:\Miners\CCminer . CPU Release .\ccminer.exe' -a verus -o
stratum+tcp://pool.verus.io: -u RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ.Donator -t
make sure you replace the pool (stratum+tcp://pool.verus.io:) with your prefered pool and the address
(RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ) with your own address. The address shown in this line is the veruscoin
foundation donation address. If desired change the identifier (Donator) to a name that easily identifies the
miner to you.
save and exit the file
run run.verushhash.bat to start mining.

The installer for Verus-Desktop suggests a standard location to install to, depending on the choice whether you want to install it for all users or just yourself.
Anyone who uses the computer (All Users)
%ProgramFiles%\Verus-Desktop
Only me:
%USERPROFILE%\AppData\Local\Programs\Verus-Desktop

Normally you don't need to worry about these locations, but in some instances you will be asked by community members providing support to look up a file in a folder in your Verus-Desktop installation.
note: changing files in these folders or subfolders may result in a corrupt installation. Only do so when instructed by our support community members.
Deamons
The daemons are located in the komodod, verusd and zcashd folders that can be found in the
resources\app\assets\bin\win\ subfolder of your installation folder.
Program settings
Verus Desktop saves it program settings on a different folder:
%AppData%\Verus-Desktop
The users settings are stored in appdata\config.json in the program settings folder.
Standard chain data and wallet locations
KMD
%AppData%\Komodo
Verus
%AppData%\Komodo\VRSC
Komodo asset chains
Any Komodo asset chain will create a subfolder in the KMD chain data and wallet folder, which is standard named. The names will be in capitals and are identical to the official asset-chain name.
%AppData%\Komodo\<CHAIN-NAME>
Note: examples
Pirate: %AppData%\Komodo\PIRATE
Utrum: %AppData%\Komodo\OOT
Zexo: %AppData%\Komodo\ZEXO
And so on...
For easy access to the binaries folders, Verus-Desktop program settings and VRSC chain folder and all binary folders, you can use the debug menu in Verus-Desktop.

First make sure your system is up to date:
sudo apt-get update && apt-get upgrade -y
I suggest not to use the root account. I you have not yet done, set up a new user
sudo adduser newusername
sudo usermod -aG sudo newusername
Switch to new username
su - newusername
Test the your new user actually has root (SuDo) access, e.g.:
sudo ls -la /root
You should get some lines like this:
drwx------ root root Jul :
Download & install the wallet binaries:
wget https://github.com/VerusCoin/VerusCoin/releases/download/v../Verus-CLI-Linux-v..-amd.tgz
The downloaded archive contains another archive and a signature text file, enabling the archive within to be verified (You'll need a running wallet to do that)
Also: Verify the URL to the latest version from the Download latest Wallet above.

```
tar -xvf Verus-CLI-Linux-v..-amd.tgz
```
Now extract the wallet archive:
```
tar -xvf Verus-CLI-Linux-v..-amd.tar.gz
```
Change directory to verus-cli
```
cd verus-cli
```
Fetch parameters, takes time, more on slow Internet connection
```
./fetch-params
```
Creating the chaindata directory
```
cd ~
mkdir -p .komodo/VRSC
cd ~/.komodo/VRSC
```
Download the block-chain bootstrap, this considerably speeds up synchronisation of the block-chain from days to minutes... (optional)
```
wget https://bootstrap.verus.io/VRSC-bootstrap.tar.gz
tar -xvf VRSC-bootstrap.tar.gz
```
Install libraries for Verus
```
sudo apt-get install libcurl g++-multilib -y
```
Install Tmux a terminal multiplexer with which you can run threads in the background see https://en.wikipedia.org/wiki/Tmux
```
sudo apt-get install tmux -y
```
Start tmux:
```
tmux
```
Launch Verus Daemon with or without number of threads
(usually number of threads equals number of cores or double of that if the processor support hyper threading well)
```
~/verus-cli/verusd -gen -genproclimit
~/verus-cli/verusd -gen -genproclimit=
```
Once mining is operational – again this may take some time –
you'll see: mega hashes complete - working
then detach tmux
[ctrl]&b d
Disable login with Root User
(make sure your newly created user login works and has sudo rights)
```
sudo nano /etc/ssh/sshd_config
```
Find: PermitRootLogin yes
And set to
PermitRootLogin no
Apply new settings:
```
sudo systemctl restart sshd
```

You can use the following scripts to automate it (prints, for each block, how many blocks remaining and the estimated date).
Keep in mind that the actual number of blocks per day is not always exactly . It varies and the estimated dates may change slightly over time. Below you will find the Linux script that will calculate the extimated time for you:
Script:
```
#!/bin/bash
#Config
verus_path='/XXXX/verus-cli/'
verus_cli='verus'
get_transactions=( $($verus_path$verus_cli listtransactions '' |grep blockstomaturity|sed 's/.e: //;s/,//g'|awk '{ print $ }'|sort -n) )
```

```
cur_block=( $($verus_path$verus_cli getmininginfo|grep blocks':|sed
's/.e: //;s/,//g'|awk '{ print $ }') )
arr_idx=''
for i in '${get_transactions[@]}'
do
days_to_mature=$(( ($i )/ ))
mature_to_date=$(date +'%m-%d-%Y' -d '+$days_to_mature days')
echo ''Block #' $arr_idx 'will mature in approximately'$days_to_mature
'days' '(' $mature_to_date ')' '
((arr_idx++))
done
```

You'll need to enter the correct path to your verus directory and you will need to have the verus daemon running in order to sucessfully run this script.

Following are some bash scriptse to help make managing your Linux-based CLI miner a bit easier. Prereq is to install mailutils (postfix) and configure with your server's FQDN and set inet_interfaces=localhost in the postfix/main.cf file.

Important General Information

VRSC Wallet & data location on Linux: ~/.komodo/VRSC

for wallet version prior to .., replace verusd with komodod.

Scripts:

MINER SRVC MONITOR & ALERT IF DOWN

Checks for the verusd daemon and if it has stopped emails you.

checkifverusdisrunning.sh

```
#!/bin/bash
if pgrep -x 'verusd' > /dev/null
then
TRUE=''
else
echo 'Merry Miner Has Stopped Mining!!! HELP!!' | mail -s 'OUTAGE: Merry'
-a 'From: user@yourqualifieddomain.tld' you@youremail.tld
fi
```

ALERT ON NEW BLOCKS MINED

Prereq: Create a file called txHistory.txt and put in it, saved to your home folder.

The script then compares the current wallet TX count and compares to the txHistory file...

so first run it will enter the right number in that file overwriting your .

Only emails you if the number changes.

checkfornewblocks.sh

```
#!/bin/bash
historicalcount=$(cat /home/user/txHistory.txt)
livecount=$(/home/user/verus-cli/verus getwalletinfo | grep txcount | sed
's/[^-]*//g')
if (($livecount > $historicalcount))
then
echo $livecount > /home/user/txHistory.txt
echo 'Merry Miner Has Mined a Total of $livecount Blocks! Woot!' | mail -
s 'Merry's Blocks: $livecount' -a 'From: user@yourqualifieddomain.tld'
you@youremail.tld
else
```

NOCHANGE=''
fi
WALLET BACKUP TO SECURED EMAIL (PROTONMAIL SUGGESTED)
For this script I recommend setting up a new Protonmail account with no association to any other service or your name, FA secure it.
Schedule script in CRONTAB
In the following, the */ is every min, the is on the hour every hour, the is every day at PM.
CRONTAB
m h dom mon dow command
*/ * * * * /home/user/checkfornewblocks.sh
* * * * /home/user/checkifverusdisrunning.sh
* * * /home/user/backupwallet.sh
Note: For any emails sent (for backup of dat file for example) make sure to enforce TLS security in postfix by adding the following line to your /etc/postfix/main.cf
smtp_tls_security_level=encrypt

Komodo is probably already running.
Procedure:
It just means you can't start it when it's already running. If it isn't running we can take a look at it.
Check to see if it is running by running:
./verus getinfo
If it's running you'll get info back, if it isn't you'll get an error.
If that throws an error, it's worth checking using this command:
ps fax | grep verus
and seeing if any processes are listed besides the process you're using to search.
You would see verusd with a child process of komodo and all its cli arguments.

Note: I use Google Cloud for my staking wallet and backups but the script works regardless of what you use; if you have timelocked coins, you will have to keep your wallet.dat file secure for a long, long time: it's crucial to maintain secure the staking instance as well as the storage..
Important General Information
VRSC Wallet and Data location on Linux: ~/.komodo/VRSC
Procedure:
If you use GCloud:
create a new directory where we'll upload the backups (ie: mkdir /home/user/backup) create a storage bucket via GCloud console (ie bucket_verus)
* mount the bucket on your local machine (syntax is: gcsfuse ; ie gcsfuse bucket_verus /home/user/backup)
The script can be crontab scheduled to run daily or you can run it manually; if you schedule it, you have to hardcode the encryption passphrase; otherwise you can input it manually each time. You will have to install gnupg to encrypt (sudo apt-get install gnupg gnupg-agent). As always: be sure to fully understand what the script does and why I left some echos to demonstration/test purposes, you can sefely remove all of them.
#!/bin/bash
Customize the SOURCE and the DEST folders

```
SOURCE_FOLDER=/home/XXXXX/.komodo/VRSC/
DEST_FOLDER=/home/XXXX/backup/
BCK_DATE=date +%Y-%m-%d.%H:%M
SOURCE_FILE=wallet.dat
DEST_FILE=$SOURCEFILE''$BCK_DATE
ENCRYPT_EXT='.gpg'
# all echo are for test purposes, feel free to delete them
echo check variable
echo SOURCE - $SOURCE_FOLDER
echo SOURCE_FILE - $SOURCE_FILE
echo DEST FILE - $DEST_FILE
echo DEST - $DEST_FOLDER
echo TO_BE_DEL - $DEST_FOLDER$SOURCE_FILE
sleep
cp $SOURCE_FOLDER$SOURCE_FILE $DEST_FOLDER$DEST_FILE
sleep
# Customize the passphrase!!
gpg --batch --yes -c --cipher-algo AES --passphrase='XXXXXX'
$DEST_FOLDER$DEST_FILE
sleep
rm $DEST_FOLDER$DEST_FILE
# Keep only last days - BE SURE to fully understand how it works, as
every 'rm' command!
find $DEST_FOLDER/wallet* -type f -mtime + -exec rm {} ;
```

Read it completely before using.
Important General Information
This guide is aimed towards Debian based Linux distributions. If you are using a different kind of distribution
(eg RPM-based, like CentOS) you will need to install the dependancies using a procedure that fits your
specific distribution.
There are active branches in ccminer github repo:
ARM (for bit ARM chips with AES intrinsic)
Verus. (standard x- pc's)
Verus.gpu (GPUs)
Note: Replace ARM in the git clone line below with the branchname above you want to use.
Procedure:
Install dependencies (specific for Debian-based distributions):

```
sudo apt-get install libcurl-openssl-dev libssl-dev libjansson-dev
automake autotools-dev build-essential git
```

For GPU-miner compilation additional sources are required (Not needed for CPU or ARM):

```
wget
http://developer.download.nvidia.com/compute/cuda/./Prod/local_installers
/cuda_.._.._linux.run
sudo sh cuda_.._.._linux.run
```

Download the source and compile:

```
git clone --single-branch -b ARM https://github.com/monkins/ccminer.git
cd ccminer
chmod +x build.sh
chmod +x configure.sh
chmod +x autogen.sh
```

./build.sh
And finally starting the miner (Change pool, address & workername to your own liking):
./ccminer -a verus -o stratum+tcp://pool.verus.io: -u RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ.donator -p x

Procedure:
These tools will help you stake on your permanently running machine using Verus CLI, instead of running the Agama GUI wallet.
Features
automatic shielding and unshielding of coinbase (staking, solo mining)
automatic tracking of total coins generated (also present in e-mail notifications)
optional e-mail notifications on newly generated coins
optional periodical wallet.dat backups, also as an encrypted ZIP archive
optional periodical wallet balance e-mail notifications
Initial Verus daemon configuration
Follow the guide in link above to set-up your daemon for the first time. You will need a transparent address, a zs address and the daemon running with correct parameters (-mint -cheatcatcher=zs...).
Do not stake with the same wallet.dat on multiple nodes.

Procedure:
In order to let your mining not interfere with other processes running on your PC, we'll need to deprioritize the mining process. This will result in your miner throttling down whenever your PC needs processing power. People have reported to be able to use CPU-heavy applications, like games, without the miner interfering, with this method.
Because the Verus wallet (GUI and CLI) does a lot more than mining, it is not recommended to use this for a solo-mining setup. This is tested on Verus NHEQminer and Verus CCMiner on Ubuntu ..
First thing you will need is pool-mining software. The link is supplied above this procedure. Download your preferred miner, extract it to your prefered location and configure the batchfile with the pool of your choice and your own mining address.
Now we'll need to adjust the batchfile to run on low priority:
NHEQMINER:
Run your miner as you normally would, with nice -n placed in front of the command you normally use,
for example:
nice -n ./nheqminer -v -l pool.verus.io: -u RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ.Donator -p x -t
make sure you replace the pool (pool.verus.io:) with your prefered pool and the address
(RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ) with your own address. The address shown in this line is the veruscoin
foundation donation address. If desired change the identifier (Donator) to a name that easily identifies the
miner to you.
CCMINER:
Run your miner as you normally would, with nice -n placed in front of the command you normally use,
for example:

nice -n ./ccminer -a verus -o stratum+tcp://pool.verus.io: -u RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ.Donator -t
make sure you replace the pool (stratum+tcp://pool.verus.io:) with your prefered pool and the address
(RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ) with your own address. The address shown in this line is the veruscoin
foundation donation address. If desired change the identifier (Donator) to a name that easily identifies the
miner to you.

When running ./verusd on a Linux distro (eg Debian or Devuan), not all dependencies may be installed by default, resulting in the errormessage error while loading shared libraries: libgomp.so.: No such file or directory or error while loading shared libraries: libz.so: No such file or directory.
To solve this you need to install the libgomp and zlibg-dev libraries:
sudo apt-get install libgomp zlibg-dev

The Linux version of Verus Desktop is supplied as an AppImage, so it does not get installed, but mounted in the /tmp folder on execution.
The folder it is mounted to will look like .mount_Verus-xxxxx where the xxxxx part will be a random set of characters, changing on every start of the wallet.
(It is a hidden folder: in order to see it, <CTRL>-H will toggle the display of hidden folders)
Normally you don't need to worry about these locations, but in some instances you will be asked by community members providing support to look up a file in a folder in your Verus-Desktop installation.
note: changing files in these folders or subfolders may result in a corrupt installation. Only do so when instructed by our support community members.
Deamons
The daemons are located in the komodod, verusd and zcashd folders that can be found in the
resources/app/assets/bin/linux subfolder of your (mounted) program folder.
Program settings
Verus Desktop saves it program settings on a different folder:
~/.verus-desktop
The users settings are stored in appdata/config.json in the program settings folder.
Standard chain data and wallet locations
KMD
~/.komodo
Verus
~/.komodo/VRSC
Komodo asset chains
Any Komodo asset chain will create a subfolder in the KMD chain data and wallet folder, which is standard named. The names will be in capitals and are identical to the official asset-chain name.
~/.komodo/<CHAIN-NAME>
Note: examples
Pirate: ~/.komodo/PIRATE
Utrum: ~/.komodo/OOT

Zexo: ~/.komodo/ZEXO
And so on...
For easy access to the binaries folders, Verus-Desktop program settings and VRSC chain folder and all binary folders, you can use the debug menu in Verus-Desktop.

Procedure:
It just means you can't start it when it's already running. If it isn't running we can take a look at it.
Check to see if it is running by running:
./verus getinfo
If it's running you'll get info back, if it isn't you'll get an error.
If that throws an error, it's worth checking using this command:
ps fax | grep verus
and seeing if any processes are listed besides the process you're using to search.
You would see verusd with a child process of komodo and all its cli arguments.

Verus Desktop
Currently only macOS ..x and up are tested and supported for Verus Desktop. Installing Verus Desktop on OSX ..x or earlier may be possible but the wallet does not function and never completes the initial sync.
Verus CLI wallet
The CLI wallet should run on lower versions without problems.
File location
VRSC Wallet and Data location on Mac OS: /Users//Library/Application Support/Komodo/VRSC
Necessary files:
Link : How-To install the latest Wallet
Procedure:
If you installed on OSX ..x or earlier and need to remove it:
Quit your Wallet.
Eject the wallet dmg.
Make a backup of wallet.dat & VRSC.conf if necessary (Only if you had a wallet on this machine or if you used an existing wallet.dat)
If you installed the Agama.app in Applications, move this to the trash.
Move ~/Library/Application Support/Agama to the trash or use the following command in terminal
rmdir /Users//Library/Application\ Support/Agama.
Move ~/Library/Application Support/Komodo to the trash or use the following command in terminal
rmdir /Users//Library/Application\ Support/Komodo.
Upgrade OS.
Install Agama (Check Link for a smooth install)

VRSC Wallet and Data location on Mac OS: /Users//Library/Application Support/Komodo/VRSC
Software needed to mine
Prerequisite: VerusCoin wallets
Needed: VerusCoin miners
Prerequisites:
To start mining with your Mac, you will need to have an address to mine to. Possibilities are:

An exchange address (NOT recommended. This is asking for trouble.)
An address from a web-wallet
An address from a mobile wallet
An address from a Verus-Desktop Lite wallet
An address from a Verus-Desktop Native wallet
Procedure:
To start mining on your Mac, you should already have a wallet and and access to an address to mine to. Now to start mining we will need to download the miner first:
. Go to the Veruscoin miners link above and download CLI nheqminer for MacOS.
. Open the download folder in finder.
. doubleclick nheqminer-MacOS-v...tar.gz to unpack it in your download folder.
. Open the nheqminer folder that appeared in your download folder
Now we need to configure things so the miner connects to a mining pool (listed on the download page, below the miner download) and mines to your address. In the instructions below I will use the VerusCoin community pool and the address for the Veruscoin foundation. make sure to use your own address and if desired the details of a different pool:
. click start.sh and open with TextEdit.app
. On the nd line, change PoolHost= into Poolhost=pool.verus.io
. On the rd line, change Port= into 'Port='
. On the th line, change 'PublicVerusCoinAddress=' into PublicVerusCoinAddress=RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ
(make sure you use your own address!!!!)
. On the th line, change WorkerName= into WorkerName=MacOS (or any name you want to identify your system)
In the next step we will set how many processor threads the miner will use. Make sure to use less threads than your processor has virtual cores. If you allow the miner to use all cores, your system will become sluggish at times, or it may even become completely unresponsive:
) On the th line, change Threads= into Threads= (fill in the amount of virtual cores you want to use)
) Save and exit the file.
We have all the settings ready to go, but the file we just edited is not executable right now.
. Click Start.sh and click copy 'Start.sh'
. Click in an empty space in you folder and click Paste items, which will result in a new start copy.sh file.
. rename start copy.sh to start.command
Now we have an executable file that will run when you doubleclick.
. Doubleclick start.command and your machine will start mining.

Procedure:
In order to let your mining not interfere with other processes running on your PC, we'll need to deprioritize the mining process. This will result in your miner throttling down whenever your PC needs processing power. People have reported to be able to use CPU-heavy applications, like games, without the miner interfering, with this method.
Because the Verus wallet (GUI and CLI) does a lot more than mining, it is not recommended to use this for a solo-mining setup.
First thing you will need is pool-mining software. The link is supplied above this procedure. Download your preferred miner, extract it to your

prefered location and configure the batchfile with the pool of your choice and your own mining address.
Now we'll need to adjust the batchfile to run on low priority:
NHEQMINER:
Run your miner as you normally would, with nice -n placed in front of the command you normally use,
for example:
nice -n ./nheqminer -v -l pool.verus.io: -u RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ.Donator -p x -t
make sure you replace the pool (pool.verus.io:) with your prefered pool and the address
(RVjvbZuqSGLGDmBBFkbHWySPKExtfjQ) with your own address. The address shown in this line is the veruscoin
foundation donation address. If desired change the identifier (Donator) to a name that easily identifies the
miner to you.

he installer for Verus-Desktop suggests a standard location to install to Applications
Anyone who uses the computer
/Applications/Verus-Desktop.app
Deamons
The daemons are located in the komodod, verusd and zcashd folders that can be found in
/Applications/Verus-Desktop.app/Contents/Resources/app/assets/bin/osx
Program settings
Verus Desktop saves its program settings in your home folder:
~/Library/Application\ Support/Verus-Desktop
The users settings are stored in appdata/config.json in the program settings folder.
Standard chain data and wallet locations
Verus Desktop saves its chain and wallet data in your home folder:
KMD
~/Library/Application\ Support/Komodo
Verus
~/Library/Application\ Support/Komodo/VRSC
Komodo asset chains
Any Komodo asset chain will create a subfolder in the KMD chain data and wallet folder, which is standard named. The names will be in capitals and are identical to the official asset-chain name.
~/Library/Application\ Support/Komodo/<CHAIN-NAME>
Note: examples
Pirate: ~/Library/Application\ Support/Komodo/PIRATE
Utrum: ~/Library/Application\ Support/Komodo/OOT
Zexo: ~/Library/Application\ Support/Komodo/ZEXO
And so on...
For easy access to the binaries folders, Verus-Desktop program settings and VRSC chain folder and all binary folders, you can use the help menu in Verus-Desktop.

Verus Wallet.dat, Chaindata & VRSC.conf standard locations
ARM Linux: ~/.komodo/VRSC
Note: Due to the size of the blockchain, on an ARM installation this directory is usually stored on a mounted external hard drive.

This will have its own file path that was specified when the drive was mounted.
Thus the path to ~/.komodo/VRSC directory is likely to be user-specified & thus non-standard on an ARM.
(NB This version is aimed at inexperienced users, so it includes the associated routine/basic Linux commands.
You will need access to the relevant directories, which may require sudo priveleges)
(NB Raspberry Pi ARM boards lack some instruction sets to mine, though the Pi & Pi are ARM & can be used as staking devices if an ARM operating system is installed)
Procedure:
Navigate to the directory which contains the verus daemon binary file (the default is in the home directory - ie ~/verus-cli)
cd verus-cli
Check the Verus daemon is running - you will need to keep it running to check the downloaded file's signature
top
(verusd should appear as one of the processes in the list. If you need to start it use the command ./verusd.)
Download the latest ARM version. Check https://verus.io/wallet/command-wallet
(The commands below are for v..-) Don't forget to change the version number to whichever one you are updating.
wget https://github.com/VerusCoin/VerusCoin/releases/download/v..-/Verus-CLI-Linux-v..--arm.tgz
unpack/unzip the download into the signature file and the binaries tar/zipped file
tar -xvf Verus-CLI-Linux-v..--arm.tgz
Check it has unpacked/unzipped & open the signature file
ls -la
cat Verus-CLI-Linux-v..--arm.tar.gz.signature.txt-file
This will display the signature information to verify the downloaded file is authentic.
CHECK the signature by using the data in the signature file
./verus verifyfile 'Verus Coin Foundation Releases@' '[signature string from signature.txt-file]' [your home directory file path]Verus-CLI-Linux-v..--arm.tar.gz
The return should be true. If you have an issue add '' around the file path.
STOP the Verus daemon
./verus stop
Check whether it has stopped running with this command. (To be doubly sure you can also look in the debug file in the)
top
(This time verusd should NOT appear in this list.)
Rename the old versions of the binaries (replace the version numbers with the version being replaced)
(This ensures you can easily revert to the old version by renaming these files.)
mv verus verus-v..-
mv verusd verusd-v..-
Check they have been renamed
ls-la

Unpack/unzip the binaries for the latest version
tar -xvf Verus-CLI-Linux-v..--arm.tar.gz
Check if they have unpacked/unzipped in the directory
ls -la
If the directory where your Verus wallet.dat, Chaindata & VRSC.conf files are stored is in a non standard place - eg on a mounted external hard drive - you may have a datadir specified in your VRSC.conf file. Navigate to that directory to check your datadir is still correct. The VRSC.conf file may have been overidden in the update. This ensures the Verus daemon knows where to find the blockchain you have currently saved on your system.
cat VRSC.conf
If it requires editing, use nano to edit this file
nano VRSC.conf
& append this to the VRSC.conf file :
datadir=[your custom file path for your mounted hard drive]
Start the Verus daemon again on the new version (add any other start options from https://wiki.verus.io/#!faq-cli/clifaq-_verusd_options.md)
./verusd
Check the version
./ verus getwalletinfo
Once you are satisfied the new version is working, clean up the older binares (from step) by deleting them (or just move them to another directory using the same mv command as step)
rm verus verus-v..-
rm verus verusd-v..-

There are minimum requirements on your hardware and software for running a Verus Wallet on ARM-devices. If your platform does not meet the minimum requirements, you may not be able to run the required software.
The listed requirements are for running one chain. Additional PBaaS chains require more resources.
Verus Wallet on CLI
Absolute minimum requirements:
-bit processor
-bit Operating system (Raspbian is standard bit)
GB memory + GB Swap available to the CLI Wallet
GB storage space for Verus Blockchain and CLI wallet
internet connectivity
libgomp and zlibg-dev libraries installed
Recommended requirements
-bit processor with AES functions enabled
-bit Operation system (Raspbian is standard bit)
GB memory or more + GB Swap available to the CLI Wallet
GB storage on a fast medium (like NVMe device) for the Verus Blockchain & CLI wallet. This supplies room for blockchain growth over time and the ability to bootstrap the wallet.
internet connectivity
libgomp and zlibg-dev libraries installed
Verus Wallet on GUI
Absolute minimum requirements:
-bit processor
-bit Operating system (Raspbian is standard bit) with GUI interface
GB memory + GB Swap available to the Verus Desktop Wallet

GB storage space for Verus Blockchain and Verus Desktop.
internet connectivity
libgomp and zlibg-dev libraries installed
Recommended requirements
-bit processor with AES functions enabled
-bit Operation system (Raspbian is standard bit)) with GUI interface
GB memory or more + GB Swap available to the CLI Wallet
GB storage on a fast medium (like NVMe device) for the Verus Blockchain & CLI wallet. This supplies room for blockchain growth over time and the ability to bootstrap the wallet.
internet connectivity
libgomp and zlibg-dev libraries installed
Staking
Absolute minimum requirements:
All requirements to run a wallet
Recommended requirements
All requirements to run a wallet
A fully configured and functioning NTP client, to keep your clock synchronized
Low latency internet connectivity
Solo Mining
Absolute minimum requirements:
All requirements to run a wallet
Recommended requirements
All requirements to run a wallet
-bit processor with AES functions enabled
Low latency internet connectivity
Pool Mining
-bit processor with AES functions enabled
-bit Operation system (Raspbian is standard bit)
ARM release of CCMiner or NHEQminer
Low latency internet connectivity
A public address to mine to
A public mining pool to connect to

When running ./verusd on a Linux distro (eg Debian or Devuan), not all dependencies may be installed by default, resulting in the errormessage error while loading shared libraries: libgomp.so.: No such file or directory or error while loading shared libraries: libz.so: No such file or directory.
To solve this you need to install the libgomp and zlibg-dev libraries:
sudo apt-get install libgomp zlibg-dev

Its the Revolutionary Layer / Protocol Launches its Most Anticipated Mainnet Upgrade
After years of hard work and dedication, on the fifth anniversary, the worldwide Verus community is excited to announce the launch of the highly anticipated mainnet upgrade on May . The Public Blockchains as a Service release marks a major milestone for Verus and the entire cryptocurrency industry.
PBaaS activation: block height ,, (Sunday May)
Turning the Crypto Industry on its Head
The wide cryptocurrency industry will see how much better Verus PBaaS is as a solution for true decentralization in cross-chain, DeFi AMMs, on-

chain PP exchange, self-sovereign provable recoverable IDs, NFT capabilities, anti-phishing, anti-MEV, scale, or just about any of the challenges people have on crypto platforms today. In fact, as we prepare for activation of this protocol that enables so many new use cases for crypto, not to mention easy onboarding when entrepreneurs discover how to really leverage VerusIDs, PBaaS is so far beyond what people are experiencing on any crypto platform today that it has been easy to dismiss our community as describing the impossible.

Once PBaaS is live on mainnet and people can actually use all of this themselves, along with every project, chain and currency on the PBaaS network and EVERY ERC or ERC on Ethereum or even bridged to Ethereum, the truth of Verus will be self-evident, and those who build on Verus will more easily build faster, better, more secure apps and services with the possibility of provable identity + privacy, crowdfunded projects, businesses, economies, and public infrastructure efforts all seamlessly integrated into UIs that do not need business deals, permission, or any centralized infrastructure to connect services and users, enabling everyone to communicate in provable, private, or public ways that always include bidirectional, secure commerce of all kinds.

Core Capabilities

The core capabilities of this release include (lots to learn to understand it all, but here's a partial list):

Verus Intersystem Protocol (VIP) — Layer Multichain and Inter-chain Protocol

To our knowledge, the VIP protocol is the only fully decentralized, provable cross-chain technology available on any network that is based on cryptographic proofs of each chain with optional witnesses to confirm chain state. From what we've been able to learn about LayerZero, Cosmos, Polkadot, Thorchain or others, VIP is different from (though closest in some ways to LayerZero), superior to, and more decentralized than all other cross-chain protocols we have seen. We can of course get into any level of discussion on the topic, but expect more descriptions and educational material to follow, now that the protocol was originally conceived of years ago, and has been developed and tested/hardened for years.

VerusID Content Multimaps — Unlimited Provable Data for Every VerusID

Of course this upgrade retains all of the groundbreaking capabilities already available with VerusID, including multisig options, separate revocation and recovery authorities for maximum security to prevent theft of identity or funds, and the ability to timelock your ID, enabling you to even thwart attempts to access your funds after someone compromised your secret keys.

With the PBaaS upgrade, VerusIDs also gain a powerful new capability that amounts to an unlimited, rent-free indexed database for every VerusID on every blockchain of the PBaaS network. This data can be encrypted or published in the clear and used for any purpose or application, whether social networking, voting, oracles, publishing any information, various forms of provable attestations, ratings, and much, much more. With the scalability of PBaaS and VerusID content multimaps together, the Verus PBaaS network becomes the most scalable, permissionless, provable source of human data from which AIs can learn about humanity in the world. We don't believe anything else even comes close, and with the recent rise of AI, it feels great to have this come together as a core original goal at a very appropriate and important time in history.

Permissionless Chain, Token, and Liquidity Basket Currency Launches
Whether you are looking to crowdfund an effort in a manner much like Kickstarter or IndieGoGo, sell identities to which you expect to add value or that are sub-IDs of a very cool root ID, or launch an entirely new blockchain economy and blockchain that starts with all of the Verus technology and a bridge to Verus and Ethereum from day , Verus is probably the best platform for you. In fact, as we were planning for the next testnet and wanted to have our own version of DAI to use in the bridge, since we want only decentralized currencies in the live mainnet bridge, we used a command to launch a token on the old testnet, exported it and all its supply to Ethereum's Goerli on the existing bridge, and that will be the DAI proxy/simulant that we will use on the new testnet. It saved us all some time, proving that Verus PBaaS will also be the easiest, most efficient way to launch even an Ethereum ERC, whether that ERC represents a token, DeFi basket, or even another PBaaS blockchain
Verus enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing % POW/% POS, % hash attack resistant blockchain that can send and receive from the Verus chain which launched it.
All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.
Verus also enables any ID owner to define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from % to %. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners, stakers, and liquidity providers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.
Decentralized Ethereum Bridge
Shortly after PBaaS activates on mainnet, the Ethereum bridge will launch as a decentralized gateway and : provably mapped currency called "vETH" on the Verus network, also available to all PBaaS chains. The Ethereum bridge will also include a % backed basket of % Verus, % Ethereum, and % DAI, which will have the following functions:
Auto conversion of fees from Verus <-> Ethereum when sending cross chain, based on the on-chain conversion price in the liquidity basket
Permissionless ability to register *.vETH IDs, which in addition to the normal sub-ID capability of creating a single token that has control over the ID and can be exported to Ethereum as an automatic ERC NFT, can alternately be used to create a "mapped currency", which can be provably mapped : to any ERC currency on the Ethereum network by exporting it to the vETH bridge. Only Verus root IDs, and IDs of a gateway can create mapped currencies to a gateway.
Decentralized, fair bridge launch. The vETH bridge will launch as other currencies, chains, and gateways can launch, including the bridge converter. When the bridge launches, the contracts will have a surplus of Verus that comes from the fees paid to launch the gateway. In the protocol, that surplus is first used to solve the chicken and egg problem

of none of one currency on the other chain. The contracts will cover the Verus fees for all sends from Ethereum before the Bridge converter launches, the liquidity basket becomes active, and the bridge recognizes that launch. Once that happens, the remainder of ½ the fees (VRSC) that are left over from the launch and did not go to miners and stakers or pay Verus fees for people sending from Ethereum.

The Verus Fee Pool and Rewards

Another technology and solution unique to Verus, the Verus Fee Pool technology goes live with PBaaS. What this means is that as people use the cross-chain, DeFi, or purchase/register IDs on the network, the miner or staker will put all fees into the on-chain "fee pool", and then take /th of the pool in addition to the block reward. This gives everyone incentive to still prioritize transactions based on fees, while preventing validators from gaming the fee system by washing fees-in vs. fees-out or working to reorg/rewind to capture a particularly juicy block reward. Instead, this technology aligns all validator incentives with the health/proper operation of the network and creates a circular on-chain economy that can last well beyond block rewards.

A Historic Move Forward

For those who actually understand the challenges actual users have in crypto today, this release represents a historic move forward in public cryptographic networks and credibly and actually neutral infrastructure for society-wide human and AI collaboration and learning. As a community, this version really does realize the vision laid out in the original Verus Vision Paper, and at the same time provides much more than was described there. We are a community of individuals, and it is only because that is what we are that we have all arrived here together. The rest of the world is stuck on their Munchausen's protocols, trying to figure out how to share the front-running and back-running spoils taken from users, but Verus activation gives them a better path forward.

As a community, we do not promise, we describe a shared vision and welcome contributors from everywhere. Now that we, as a community, have delivered on the original phases described in the first vision paper together, it is time for each of us to also consider what we might personally want to build or be part of building over this incredible, geoscale, ID-enabled, fully decentralized platform. Our next efforts across the community should be to realize the promise of this network in the use cases we create, use cases that we will have an advantage creating on Verus over any other platform because it really is that much better.

Open-source, Rent-free, Scalable Public Infrastructure

Verus is an open-source decentralized blockchain protocol with proof-of-work and proof-of-stake as its consensus mechanism. It offers rent-free blockchain tools for creators and organizations to build products, services and systems.

Verus is a multichain protocol with strong focus on scalability, security and decentralization. It can scale to world demand, is proven % hash attack resistant and is community built - all coins in circulation are fairly mined and staked.

With Public Blockchains as a Service (PBaaS) anyone can launch scalable, interoperable, customizable and independent blockchains for public or private use. Create tokens and currencies on top of blockchains for any

use case. Get funding and create markets with protocol level built-in DeFi.
Protocol-level self-sovereign digital identities and namespaces are at the center of the Verus multichain protocol. Giving builders, communities, businesses and organizations tools never before seen.
Everything happening on the Verus multichain protocol has aligned incentives with the miners and stakers of the worldwide network. This makes it one of the most secure protocols with opportunities to earn.
#Accessibility
Verus creates blockchain technology that is accessible for everyone.
No programming needed for blockchain, token and currency launches
Low protocol fees
Mining and staking accessible for everyone
#Scalability
Verus achieves practically unlimited scalability through its Public Blockchains as a Service.
Create use case specific blockchains and currencies
Move activity between chains to avoid congestion
#Security
Verus takes blockchain security to the next level.
% hash attack resistant through Verus Proof of Power
#Interoperability
Verus enables a world where all blockchains communicate with eachother.
Cross-chain, cross-system transfers of all data types
Bridges to communicate with external blockchains
#Community
Verus is a community driven project in the true spirit of Bitcoin.
Fair launch
No ICO, no developer fees, no founder fees, no premine
Decentralized public blockchain

Coin Overview
Launch Date May
Coin Ticker VRSC
Average Block Time minute
Transaction Fee . VRSC
Max Supply ,, VRSC
Consensus Algorithm Verus Proof of Power
Hash Algorithm VerusHash .
Privacy Zcash Sapling

Verus Proof of Power
VerusPoP is a % proof-of-work, % proof-of-stake consensus algorithm. More information on the Verus miner and staker ecosystem.
#Hybrid Consensus
Verus Proof of Power, or VerusPoP, is a hybrid consensus algorithm which uses a statistical function that combines Proof of Work (PoW) and Proof of Stake (PoS) to validate each block by either PoW or PoS, while averaging to a target percentage of blocks being validated by each form of proof.
In short, it a unique consensus mechanism with % of all blocks validated by miners, and the other % by stakers.
#Attack Resistant

To successfully attack the Verus blockchain, more than % of the validation power is needed, called Chain Power. A % attack would require a combined value of over % of both the chain’s hashpower and its coin supply. For technical information on VerusPoP read the whitepaper (opens new window).
VerusPoP provides a decentralizing effect on the network, incentivizing holders to keep nodes online to support the network. Even if a change in network hashrate happens, the PoW/PoS ratio stays the same: /%.
#VerusHash .
From the VerusPoP whitepaper (opens new window):
'VerusHash is specifically developed to deliver a competitive advantage for CPUs with GPUs. It is an exceedingly CPU-friendly long input hash function that uses the quantum-secure, short input Haraka V as its core compression algorithm. The result is the fastest known cryptocurrency hash algorithm available to modern CPUs and the only hash algorithm which enables today’s CPUs and GPUs to compete on an economically comparable level.
Haraka V is designed as a short input hash to exclusively consume one chunk of bits and produce bits of a hash result. Utilizing Haraka V VerusHash takes any length of input and produces a bit hash result, unique to VerusHash, that also provides the same security guarantees as Haraka V. This makes VerusHash bit secure for classical computing attacks and bit secure against quantum computers for pre-image and second pre-image attacks.
To understand the VerusHash algorithm it helps to first separate the digest from the core. We then consider the Haraka V core as an abstract digest function that takes bits (bytes) of input and produces bits (bytes) of output. Given such a digest function, referred to as haraka, the most concise implementation of VerusHash, in any language to-date, is the following Python code for the VerusHash hash digest as follows:'

```
# verus_hash
def verus_hash(msg):
buf = [] *
length = len(msg)
for i in range(, length, ):
clen = min(, length - i)
buf[:] = [b for b in msg[i:i + clen]] + [] * ( - clen)
buf[:] = haraka(buf)
return bytes(buf[:])
```

#PoS Problems Solved
Verus’ staking algorithm solves the two major theoretical issues undermining other PoS systems, Nothing at Stake and Weak Subjectivity by leveraging its smart transaction capabilities to remove any incentive to attempt cheating, making it a losing proposition. Read: How Verus Solved Proof of Stake’s Two Biggest Problems: Nothing at Stake and Weak Subjectivity

Verus is a decentralized public blockchain, a community driven project in the true spirit of Bitcoin. Anyone can participate and contribute, no matter who you are or where you come from. Verus is:
Open
Borderless
Public
Neutral

Censorship Resistant

#Fair Launch

Verus had a fair launch, meaning that everyone had, and still has equal opportunity to collect its currency through mining and staking. For Verus this means:

No ICO has been held
No founder or developer fees
No premine
No commercial interests
No rent-seeking behavior

Launch without programming needed

Enables any user with VerusID to create their own token currency or even full fledged, multi-currency, VerusID-issuing % PoW/% PoS, % hash attack resistant blockchain that can send and receive from the Verus chain which launched it.

All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, VerusID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

#DeFi liquidity pools and fractional currency baskets

Any VerusID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from % to % backing.

The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

#Bridge converter launches

Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain.

Bridge converter currencies have the same flexibility as other fractional % asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross-chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

#Crowdfunding currency and blockchain launches

Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees.

The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

#Interoperable, multichain network

The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access.

There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Defining a currency

There are many options to choose from when defining your currency or blockchain. Combine them in the options parameter for different use cases.

Options Details

OPTION_FRACTIONAL Allows reserve conversion using base calculations when set

OPTION_ID_ISSUANCE Clear is permissionless, if set, IDs may only be created by controlling ID

OPTION_ID_STAKING All IDs on chain stake equally, rather than value-based staking

OPTION_ID_REFERRALS If set, this chain supports referrals

OPTION_ID_REFERRALREQUIRED If set, this chain requires referrals

OPTION_TOKEN If set, this is a token, not a native currency

OPTION_SINGLECURRENCY For PBaaS chains or gateways to potentially restrict to single currency

OPTION_GATEWAY If set, this routes external currencies

OPTION_PBAAS This is a PBaaS chain definition

OPTION_GATEWAY_CONVERTER This means that for a specific PBaaS gateway, this is the default converter and will publish prices

OPTION_GATEWAY_NAMECONTROLLER When not set on a gateway, top level ID and currency registration happen on launch chain

OPTION_NFT_TOKEN Single satoshi NFT token, tokenizes control over the root ID

Need help setting up a currency launch?

Go to the Verus Discord #pbaas-development channel. The community is happy to assist!(opens new window)

#VerusID Namespace

To create a currency (or blockchain) of a specific name, you need a VerusID of the same name. The controller of this VerusID is the only one who can create a currency of that name, and they can only do so once.

#Example scenarios

So, let's hypothetically assume we have VerusIDs, one named gold@, one named mycoin@, and one named bob@.

We would like to have one currency, gold@, that we somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.

We also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, 'my', offers to attribute the coins some utility or product value if the purchase exceeds a certain level.

#Defining GOLD currency

Of course, since this is a test currency, we send ourselves some to start. We could define the currency gold as follows:

```
./verus -chain=VRSCTEST definecurrency '{
'name':'gold',
'options':,
'currencies':['vrsctest'],
'conversions':[.],
'minpreconversion':[],
'preallocations':[{'david@':.}]
}'
```

The identity of the currency must be funded with at least VRSCTEST before sending the transaction returned from this command to initiate a currency launch that will start at blocks from when it was made (default), and that must have VRSCTEST preconverted at . VRSCTEST per GOLD in order to launch.

All of this happens as part of the mining process, since mining the blocks that launch a currency earn the .% conversion fees of participation in the launch, converting VRSCTEST to GOLD.

#Converting VRSCTEST to GOLD

We could send the following command before the block where GOLD token launches. After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, we could issue the command:

```
./verus -chain=VRSCTEST sendcurrency '*' '[{
'address':'bob@',
'convertto':'gold',
'preconvert':,
'amount':
}]'
```

This would effectively park our conversion until the token launches, at which point, we will either find . GOLD in our wallet, or we will have our VRSCTEST back.

Assuming it launches, and we later want to create MYCOIN, which can be converted to with either GOLD or VRSCTEST.

#Defining MYCOIN currency

Let's define MYCOIN with:

```
./verus -chain=VRSCTEST definecurrency '{
'name':'mycoin',
'options':,
'proofprotocol':,
'currencies':['vrsctest', 'gold'],
'minpreconversion':[,.],
'initialsupply':
}'
```

#Minting new MYCOIN

In 'mycoin', we set proofprotocol to , which is PROOF_CHAINID. That means that the controller of the mycoin@ VerusID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency 'mycoin@' '[{
'address':'bob@',
'currency':'mycoin',
'mintnew':,
'amount':
}]'
```

Defining a PBaaS-Blockchain
There are many options to choose from when defining your blockchain. Combine them in the options parameter for different use cases.
Options Details
OPTION_FRACTIONAL Allows reserve conversion using base calculations when set
OPTION_ID_ISSUANCE Clear is permissionless, if set, IDs may only be created by controlling ID
OPTION_ID_STAKING All IDs on chain stake equally, rather than value-based staking
OPTION_ID_REFERRALS If set, this chain supports referrals
OPTION_ID_REFERRALREQUIRED If set, this chain requires referrals
OPTION_TOKEN If set, this is a token, not a native currency
OPTION_SINGLECURRENCY For PBaaS chains or gateways to potentially restrict to single currency
OPTION_GATEWAY If set, this routes external currencies
OPTION_PBAAS This is a PBaaS chain definition
OPTION_GATEWAY_CONVERTER This means that for a specific PBaaS gateway, this is the default converter and will publish prices
OPTION_GATEWAY_NAMECONTROLLER When not set on a gateway, top level ID and currency registration happen on launch chain
OPTION_NFT_TOKEN Single satoshi NFT token, tokenizes control over the root ID
Need help setting up a blockchain launch?
Go to the Verus Discord #pbaas-development channel. The community is happy to assist!(opens new window)
#VerusID Namespace
To create a blockchain of a specific name, you need a VerusID of the same name. The controller of this VerusID is the only one who can create a blockchain of that name, and they can only do so once.
#Examples of blockchain launches
#Blockchain
```
./verus -chain=VRSCTEST definecurrency '{
'name':'PBaaSChain',
'options':,
'currencies':['VRSCTEST'],
'conversions':[],
'eras':[
{
'reward':,
'decay':,
'halving':,
'eraend':
}
],
'notaries':[
'Notary@',
'Notary@',
'Notary@'
],
'minnotariesconfirm':,
'nodes':[
{
'networkaddress':'...:',
```

```
'nodeidentity':'Node@'
},
{
'networkaddress':'...:',
'nodeidentity':'Node@'
}
],
'gatewayconvertername':'Bridge',
'gatewayconverterissuance':
}'
'{
'currencies':['VRSCTEST','PBaaSChain','USD'],
'initialcontributions':[.,,],
'initialsupply':
}'
#Blockchain
./verus -chain=vrsctest definecurrency '{
'name':'v',
'options':,
'currencies':['vrsctest'],
'preallocations':[
{
'allnotary@':
}
],
'conversions':[],
'eras':[
{
'reward':,
'decay':,
'halving':,
'eraend':
}
],
'blocktime':,
'idregistrationfees':,
'notaries':[
'allnotary@',
'allnotary@',
'allnotary@'
],
'startblock':,
'minnotariesconfirm':,
'nodes':[
{
'networkaddress':'...:',
'nodeidentity': 'allnotary@'
},
{
'networkaddress':'...:',
'nodeidentity':'allnotary@'
},
{
'networkaddress':'...:',
```

```
'nodeidentity':'allnotary@'
}
],
'gatewayconvertername':'Bridge',
'gatewayconverterissuance':
}'
'{
'currencies':['VRSCTEST','v'],
'initialcontributions':[,],
'initialsupply':
}'
```

Defining a currency with a : mapping of an ERC-

When defining a currency it can be mapped to an ERC- :. The currency on Verus and the ERC- on Ethereum are then always interchangeable to .

↔ Verus-Ethereum Bridge

The Verus-Ethereum bridge is the first of its kind. All tokens and currencies flowing over the bridge are never in anyone’s custody, and are proven and verified by consensus rules.

☞ Access the Verus-Ethereum Bridge (opens new window)(Goerli testnet)

#Defining the currency

To create a currency of a specific name, we need a VerusID of the same name. The controller of this VerusID is the only one who can create a currency of that name, and we can only do so once.

The cost for a VerusID on the Verus testnet is VRSCTEST (or when using a referral). The cost to launch a currency is VRSCTEST. Before launching we need to have enough VRSCTEST in the namespace VerusID.

In our example we have a namespace MyUSDC with which we want to launch a currency that is mapped to the Ethereum USDC ERC- (on Goerli testnet, see contract address (opens new window)).

Below is the command to map a currency : to an ERC- on Ethereum. The address field is the Ethereum smart contract address of the ERC- we want to map to.

```
./verus -chain=VRSCTEST definecurrency '{
'name':'MyUSDC',
'options':,
'systemid':'veth',
'parent':'vrsctest',
'launchsystemid':'vrsctest',
'nativecurrencyid':{
'type':,
'address':'xDCBadccCbfBa'
},
'initialsupply':,
'proofprotocol':
}'
```

After we put in the command, we get returned a HEX. We use this HEX to launch the currency on the network. Use the command below to launch the currency:

```
./verus -chain=VRSCTEST sendrawtransaction 'HEX'
```

Now we have to wait a few blocks for the currency to be available on the network.

#Export to Ethereum

The last step is to export the currency to Ethereum so we can see it there too.

Export a currency to Ethereum (as ERC-)
A currency on Verus (testnet) can be exported to Ethereum as an ERC-. The currency can then be used in the complete Ethereum ecosystem, and on the Verus network. Sending to and from Verus and Ethereum couldn't be easier.
↔ Verus-Ethereum Bridge
The Verus-Ethereum bridge is the first of its kind. All tokens and currencies flowing over the bridge are never in anyone’s custody, and are proven and verified by consensus rules.
☞ Access the Verus-Ethereum Bridge (opens new window)(Goerli testnet)
#Exporting the currency
Now, let’s export a currency from Verus to Ethereum as an ERC- over the non-custodial bridge. We must have enough funds to pay for the export.
The export fee is: ~ . vETH or ~ VRSCTEST
The command to export a currency to Ethereum as an ERC-:

```
./verus -chain=VRSCTEST sendcurrency 'myVerusID@' '[{
'address':'xADCEEFBcbCefBDEDADBE',
'currency':'myCurrency',
'amount':,
'exportto':'veth',
'exportcurrency':true,
'feecurrency':'veth'
}]'
```

Let's break the command down.
myVerusID@ is the from- and change-address. The fee to pay for the export comes from here, and if you paid too much fees the rest will be returned here.
address is the Verus-Ethereum smart-contract bridge address. Don't change this.
currency is the name of the currency you wish to export as an ERC-.
feecurrency is the currency the export fees are paid in. Omit this to pay the fees in VRSCTEST.
Wait for notarization
After the bridge has been notarized to the blockheight where you have exported the currency, you can choose it from the token dropdown on the bridge website

Verus is a rent-free blockchain protocol that has a decentralized economy. Costs for services paid to the protocol are distributed to the miners and stakers. There are no developer fees, and no rent-seeking profiteers.

Verus has a naturally decentralized miner ecosystem. VerusHash . is the mining algorithm that is specifically developed for Verus. The algorithm equalizes mining fairness across hardware classes.
To mine Verus no specialized mining equipment is necessary. Anyone can start mining with consumer-grade hardware. Mining is made especially easy for starters, since they can start directly from within Verus Desktop, without any additional software requirements.

Additionally, when PBaaS goes live on mainnet, miners can choose to mine up to PBaaS-chains simultaneously, without losing any of their original hashing power

Anyone can start staking Verus, even with the smallest amount possible, e.g. . VRSC. Although it could take many years before you win a block with such a tiny amount.
Verus has a powerful and fair staking system. No threshold required and no registrations necessary.

Miners and stakers are rewarded for their efforts to keep the network secure. They earn block rewards. The Verus block reward emission schedule:
Block Height (& Start Date) Block Rewards Coin Emission
(..) - , VRSC
, (..) * ,, VRSC
, (..) * ,, VRSC
, (..) ,, VRSC
, (..) ,, VRSC
, (..) ,, VRSC
,, (..) ,, VRSC
,, (..) ,, VRSC
,, (~..) ,, VRSC
,, (~) . ,, VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
,, (~) . , VRSC
etc. ... ...
The maximum circulating supply of all coins is ,, VRSC.
* , Block Rewards □
From block height , on, all block rewards of and VRSC were timelocked and then unlocked at random block heights. The last of those rewards were unlocked at block height ,,.

Fees are generated from users paying for services and interacting with the protocol. These fees are processed in the blocks and earned by miners and stakers.
Protocol Activity Fees*
VerusID Registration mainnet - VRSC
PBaaS-chain Launch testnet , VRSC**
Currency/Token Launch testnet VRSC
Conversion Fee testnet .%
Transaction Fee . VRSC
* Testnet fees are subject to change. **Half the fee goes to the miners and stakers of the newly launched PBaaS-chain.

All the fees generated by the protocol are collected in the fee pool. With each new block, % of the fee pool is added on top of the block rewards. It is expected that fees outnumber block rewards, making mining and staking in the Verus ecosystem a profitable and competitive endeavor.

Mine solo through Verus Desktop to receive full block rewards. Depending on hashrate it may take a while before winning a block. No additional software is needed to get started

Participate in pool mining to receive regular rewards. You will need to set up a few things before you can start.

Solo mining Pool mining
Full Node yes no
Regular Rewards no yes
Setup Difficulty easy intermediate

Mine Verus with various devices. Profitability indication means the electricity usage vs hashrate.
Device Profitability Indication
CPU (processor) high
GPU (graphics card) medium
Mobile Phone high
ARM (not RP) high
FPGA not possible
ASIC not possible

You would need a software and a pool url to mine veruscoin, Some of the softwares that you can download is NheqMiner from veruscoin github organisation, CCMiner by Monkins or SRBMiner.

For mining with your GPU you would need CCMiner GPU miner by Monkins.

Pool Name Fee %
Verus Pool (opens new window)Fees donated to the Verus foundation %
LuckPool(opens new window) %
ZergPool(opens new window) .%
CiscoTech(opens new window) %
LePool(opens new window) %
Zhuaao(opens new window) %
AlphatechIT(opens new window) .%
Wattpool(opens new window) .%
Data(opens new window) .%

Start mining with your phone.
Download VerusBox Download on Google Play Or
VerusMiner APK Download

Staking is accessible for everyone. Use Verus holdings to secure the network. It does not matter how much $VRSC you have, there are no minimum requirements to start staking.
On the Verus blockchain UTXOs (unspent transaction outputs) are staking, not balances. A large UTXO has more chances of winning a block than a small UTXO.

There are a few rules you need to know before you can start staking.
Rules
Wallet Running
Full Node Required (native mode in Verus Desktop)

Staking Enabled
UTXO Eligible After Blocks
Minimum of . VRSC in Wallet

community operated staking pools are available.
HIGH RISK
When joining staking pools, you give away custody of your coins. You don't own your coins anymore and will have to trust the pool operator. Do at your own risk.

Imagine a new world. Everything you value is safe and secure. Everything you do, see, say and hear can be verified if you choose. Everyone you meet can present a verifiable reflection of the sum of their lifetime experiences. A society where your values, votes, and voice can be immutably expressed and secured by your community. All your data is verified, private, protected, and provable in abling exchanges of self-evident truths. A society where equity of opportunity is available to all. Kindness can be exchanged without fear of fraud or humiliation. All exchanges can be certified, smart, and secure without the need to rely on costly intermediaries. Imagine a worldwide network as scalable and as accessible as today's internet. Capable of processing financial transactions and dollars, euros, Bitcoin, Ethereum, all major currencies. Even custom currencies or auto-rebalancing multi-currency baskets, defined by any organization or individual. This network is not run by any single company or government. Instead, it's a network of independent networks. All connected by a strong cryptographic protocol that ensures all funds of all currencies are accounted for flawlessly. And by doing so, enables anyone in the world to contribute to powering it while enhancing its security and take a reward for doing so. Not only can anyone participate in this network by helping it run, the network makes it easy for anyone worldwide to create new fractal subnetworks that are launched from the network. And after launch, are completely independent from, yet fully connected and interoperable with all other parts of the network. With each new community, country, company or cause that creates its own subnetwork, the network of networks does not congest. But instead, it gains the independent processing of a new subnetwork along with its additional scale and power. Imagine that all the network could be accessed and used by people, organizations, businesses and governments easily, securely, and without any complex programming. You've probably already guessed that I'm not talking about Bitcoin, which is a decentralized tour to force, but only a single currency network. I'm also not talking about any near-term evolution of Ethereum or Ethereum-like networks, which are fundamentally single currency networks as well. Yes, I did just say that Ethereum is a single currency network. That may surprise some people so let me explain. ETH is in fact the only currency that has provable accounting as part of Ethereum's blockchain protocol. It's only contracts. Code, written, copied, and or modified by many, many separate devs and companies, interacting on a simulated computer, the Ethereum virtual machine. All of which hopefully follow best practices and conventions that enable Ethereum to transact in currencies besides ETH itself. The issue is that these contracts define currencies and engage in currency transfers between users and each other without any network-wide protocol that verifies those transfers correct accounting. All contracts may also expose arbitrary functions to other contracts

while operating together in what is called a touring complete virtual machine, meaning they can effectively do anything, or make any mistake that can be written into code. Defining accounting for and transacting in any currency besides ETH on the Ethereum blockchain as all similar VM-based blockchains is a matter of how multiple contracts interact, a black box operation to the protocol which cannot be verified by design. That architecture invented by Ethereum years ago and now used by countless protocols, leads to other risks as well, such as multi-step approval and spend operations of non-native currencies, which are quite complex for users and can expose them to fishing scams that request permission to take assets under the guise of performing another action. This new form of fishing attack on users of the most popular centralized NFT marketplace recently took about $ million from unsuspecting users, but total losses from contracts far exceed millions on a regular basis. Estimates of losses to hackers from these contracts not including fishing. In just the last year, range from . to over $ billion. All of that on blockchain networks running a similar VM-based architecture as Ethereum. Perhaps it's time for a better design. Imagine a multi-blockchain network with a real multi-currency protocol that verifies all transactions in all currencies as securely as Bitcoin and Ethereum verify transactions that transfer BTC or ETH. The good news is that now you can do more than just imagine. The system I've described is fully functional today on the various test net and being prepared by contributors across the community now for its imminent main net upgrade. All technologies are currently available for application development by those early adopters who understand the opportunities in leveraging a new disruptive paradigm. It's still officially a test net, but is fully decentralized, functional, and accessible worldwide for launching test currencies, independent and connected blockchains, on-chain exchanges of currencies, and contracts, and more. Various itself is not a new cryptocurrency. In fact, it's an independent blockchain launched May , , with no ICO, no pre-allocation of funds, and no development fee or rent built into the protocol whatsoever. Various is complete with both transparent and zero-knowledge privacy protocols, and in ecologically efficient, provably % hashtag resistant consensus protocol. It's an evolution of knock-a-modo consensus that incorporates proof of stake as well, and the only blockchain with self-sauvrent, revocable, recoverable, and interoperable addresses, and identities built into the core protocol. Various identities are supported across all transaction types, compatible with other provable ID technologies, and serve as a foundation for all currencies and blockchains on the network. Applications of the Various Network extend far beyond D-Apps of blockchain today. Any individual, project, organization, or community, needing an easy-to-use, reliable, decentralized, rent-free currency, fundraising platform, voting network, automated accounting system, or self-sauvrent, revocable, recoverable identities, and a governance model can connect to the Various Internet of Value and get started with no programming required. Solutions that used to take weeks or longer to design and code with solidity experts can now be done better, faster, and more securely, no programming needed. With Various's multi-chain model, scale challenges of today can be naturally solved by scaling out as needed, whether requirements call for centralized corporate, open-government, or fully-decentralized applications. In addition to this new technology, disruptive even among modern blockchains, the open community-driven genesis and approach

enables Various to be the first, and only rent-free, decentralized financial network capable of hosting multiple independent, yet connected real-world economies. With thousands of connected blockchains, millions of fungible currencies, and billions of users. Various is an open protocol, implemented in fully open source, designed to connect, interoperate with and serve as either independent or augmenting infrastructure for existing, or completely new, and more scalable, provable, fair, and trustworthy systems than what we have anywhere today. You can download the free and open-source Various Wallet from Various.io. Connect to the public test net, drop by the Discord for some free test currencies, and get started. Without any actual programming, just commands, you will be able to create revocable recoverable identities that you can then use to launch currencies, blockchains, liquidity baskets, automatically bridged ERCEAR, Ethereum currencies, and multi-chain applications using all currencies available on Various or Ethereum.

Unspend transaction output, or in short, UTXR. Let's visualize a part of a blockchain. We fill three of them up and leave a fourth new block empty for some new transactions. This is Alex. Alex has a various wallet and has received various coins so far, in four separate amounts. Now, these four separate amounts is essentially what UTXR is all about. Let's use two balls to explain. The first one shows the total amount of coins in your wallet, pictured is water in the bowl. The second one shows four water capsules in their bowl, each holding their own amount. They don't mix in the second bowl. Now let's have a look with that means. Meet Mary, a friend of Alex. She recently created a various wallet, but does not have any coin share. So Alex wants to send Mary of his coins to get her started. The various wallet will try to use the smallest UTXR values first in order to fill a desired coin transaction size, which in this case means it needs all four UTXRs to get to . When Alex sends the transaction, all four UTXRs will be spent and turns them into STXRs, spent transaction outputs. This is just a background process required and you will never see it. Next, two new transactions are created on the blockchain, for two new UTXRs, one for the various going to Mary, and one for sending the change of the transaction back to Alex. The transactions will go into the latest block in the blockchain so they can be confirmed by all the notes. I will explain how that all works in another animation. So all that is left to do is to send the various coins to Mary's wallet and send the change of the transaction back to Alex's wallet, and this of course is all done automatically.

Verus is a community project that was fair launched May , . No ICO was held, no coins were pre-mined, and there are no founder or developer fees. Varus' history comes by way of Bitcoin, Zikaash and Komodo, building some of the foundations of blockchain, privacy, the lead proof of work and smart transactions. Varus' consensus algorithm is called proof of power. Proof of power is a split of % proof of stake and % proof of work using Varus' hash, which is optimized for CPU mining, giving everyone a fighting chance on a level playing field. Varus' proof of power is verifiably % hash attack resistant. Varus' ID is the first self-sovereign decentralized identity system of its kind. Firstly, it takes your standard long and impossible to remember address and allows you to create a friendly name, such as John Smith, that you can use to send and receive funds. Your name can include almost all Unicode characters,

including emoji. If you ever lose access to your private keys or they become compromised, another ID can be used to securely revoke and recover your name and the fund stored on it. Along with these great features of Varus' ID, you can use the ID to create verifiable digital signatures, decentralized attestations, send and receive private messages, and more. Varus' vault gives users the ability to have a custom defined time lock of their assets that they can still stake with or revoke and recover if needed. Another great feature of Varus is the public block chain as a service, or PVAS, giving anyone the ability to launch tokens or full-interoperable block chains with no programming needed. You can set the price for identities on your chain as well as coin supply and emission, and it comes with all of the features that Varus offers, making it easy for anyone to start a project on the Varus platform, or to start their own block chain with a few clicks of a button. Half of the fee for block chain creation goes into the fee pool to be paid to miners and stakeers over time, and the other half goes into the fee pool of the newly created block chain. An exciting feature of PVAS creation is the ability to do crowdfunding launches, where you can set a minimum collection for that currency to launch. Offer pre-launch participation discounts, pre-conversion currency carve-outs, or price-neutral launch pre-allocations, all with the ability to automatically refund if the fund raise in conditions are not met. PVAS chains are fully interoperable, with cross-chain spends and currency conversions, Varus and PVAS chains can also be merged mine together, up to chains at once with all your hash power securing every chain. This helps small projects get hash power to secure the network for a lower cost. The first bridge is a decentralized bridge to Ethereum for th and ERC tokens, with other cross-platform capabilities in the future. This opens up the Varus ecosystem to value from other block chains, makes tokens and coins from the Varus network available on other platforms, and allows the storing of assets from other chains in a various vault protected Varus ID. The benefits of having layer defy on a UTXO blockchain is the ability to have consensus and reward models that are aligned with the economic activity happening on the chain. Varus uses a parallel processing model that solves all defy transactions in a block at the same time, and with the same pricing for all trade volumes and directions in that block. This, combined with the feeple, provides minor extracted value resistance, increased security, and conversion pricing that is fair and more stable than other systems. Find out more at Varus.io.

Our internet and economies have become globally centralized. Our data, bought and sold to the highest bidder. Media platform sensored in the technology engineer to funnel and extract value and commodifying control our behavior. With the emergency blockchain technology, we have a chance to develop new systems in engineer than for more sovereignty, privacy, and freedom. But many blockchain projects fall prey to the same patterns of centralization. Built in silos, they lack scalability and few teams build bridges to work between them. Now, with this understanding, we have a better way forward. The next evolution in decentralized tech is here. Varus is a truly free, open-source blockchain protocol designed for privacy, safety, open participation, and unlimited scalability. Varus is not a business or bank. To ensure integrity and impartiality, the secure public network is supported by a worldwide community of contributors, including a nonprofit foundation. Varus combines the latest in

cryptography and zero-knowledge privacy to enable its vision of public blockchains as a service. Decentralized financial and communications tech that can scale to a new internet of value and data exchange. With Varus, any individual or organization can create recoverable self-sauver identities, own their data and build and run apps and business software on any PC or mobile platform. In coming soon on Varus, individuals and organizations can tokenize assets and create currencies, create a shared economy by tokenizing a house for example, or bootstrapping crowd-funded project with fully liquid algorithmic conversion across multiple blockchains and crypto currencies. Individuals and communities can create their own micro economies and connect to a worldwide network about their public blockchains, organizations, and economies, and they don't have to pay anyone to do these things or give away their rights to their data. Now is the time to build. These are the tools we need to do that. Build with us. Varus. Truth and privacy for all. Learn more and join the conversation on Discord.

etwork Upgrade

This is the first release candidate (RC1) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.
This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today's centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate. This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.
2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:
    - **Primary -** this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.
    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.
    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.
3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.
4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@"`.
5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID
- 20 VRSC to the referrer’s referrer’s ID
- 20 VRSC to the referrer’s referrer’s referrer’s ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
That will respond with something like:
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use...

Assets 8

v0.5.9-24

Asherda
v0.5.9-24
89d2db0
v0.5.9-24 Pre-release

This release is intended for use in the VRSCTEST network. Mainnet use is still being tested.

Notable Changes

- Add blockchain rule to consider any identities without primary addresses or nonzero minimum number of signatures as invalid.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign,

revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.
2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:
    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.
    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.
    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.
3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.
4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using "z_getbalance identity@".
5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ '[{"address":"alice@", "amount":100}]'
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the

same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID
- 20 VRSC to the referrer’s referrer’s ID
- 20 VRSC to the referrer’s referrer’s referrer’s ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
That will respond with something like:
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may

display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/5cc801f027d86a59d4d70de91697fa86d96ca5034f957c81c0a2f7895a5a5677/detection
Linux-AMD64:
https://www.virustotal.com/#/file/0a989ab93e059fc778475498459b9de0484557ee8f44feb2fe418c58d976c38a/detection
Windows:
https://www.virustotal.com/#/file/2a661915722798ba65ca84990707c4f6cea1f390a79f717adc83c8b50861b581/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.5.9-22

Asherda
v0.5.9-22
`4ab9f03`
v0.5.9-22 Pre-release

This release is intended for use in the VRSCTEST network. Mainnet use is still being tested.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.
2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:

    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.
    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.
    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.
3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.
4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using "z_getbalance identity@".
5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ '[{"address":"alice@", "amount":100}]'
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name "front running", you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity.

With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer's ID
- 20 VRSC to the referrer's referrer's ID
- 20 VRSC to the referrer's referrer's referrer's ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
That will respond with something like:
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the command `listidentities`, and you may display specific identities with the command `getidentity`. If you have the authority, or use identities under your control as revocation or recovery authorities for other identities, you may also revoke and recovery identities as well.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/23ae5a14f89a0806f1a6ac143c01a2a46bc3ddbe39136534444824e02b94329c/detection
Linux-AMD64:
https://www.virustotal.com/#/file/82bf9054feaf39e750a98542086bf5e5217ce18200bd8ff40e509bbb4db00ac3/detection
Windows:
https://www.virustotal.com/#/file/e18d521cf928025dc935a6b280909cd4aaf06ed77fb60a0af5f43f92ddeb2805/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.5.9-2

Asherda
v0.5.9-2
`fff5015`
v0.5.9-2

IMPORTANT SECURITY/PRIVACY UPDATE

Version 0.5.9 incorporated a fix for the Zcash protocol issue described in:
https://z.cash/support/security/announcements/security-announcement-2019-09-24/

Updating to v0.5.9 or more recent version is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Fixed address parsing in getrawtransaction

Testnet Reset

The testnet was deleted and relaunched on the 0.5.9 release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO 0.5.9, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol

.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1

conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between

the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/13dbc2ac7d64f5bf507187b343359c6afc082138bd37d27ae42180b1e5150e49/detection
Linux:
https://www.virustotal.com/#/file/d6e6b789457ba789bc98385bc444d16c2d8d56f1fe04beeb6fe2bdfadb36b664/detection
Windows:
https://www.virustotal.com/#/file/c0490311446822c8667a202cad64d90539a45ed84da8a90d7d616df4bbcfe995/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false p...

Assets 8

v0.5.9-1

Asherda
v0.5.9-1
`1afe8b7`
v0.5.9-1

IMPORTANT SECURITY/PRIVACY UPDATE

Version 0.5.9 incorporated a fix for the Zcash protocol issue described in:
https://z.cash/support/security/announcements/security-announcement-2019-09-24/

Updating to v0.5.9 or more recent version is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Avoid re-indexing whenever possible

Testnet Reset

The testnet was deleted and relaunched on the 0.5.9 release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO 0.5.9, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol

.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/341d9d06eed1c8df0057d56f853667e9738e8ec89cd6329a84c81543d38582fe/detection
Linux:
https://www.virustotal.com/#/file/eaf521b19ff4c6d8bfc84d25a59ae6ac745003f3927ed3fc71d5a6af883b32c7/detection
Windows:
https://www.virustotal.com/#/file/10e6bb5be7b7aa33bb57beb6105a5477e8f863c8f3d2b487f239ede9864dbbfc/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives...

Assets 8

v0.5.9

Asherda
v0.5.9
9e4bb76
v0.5.9

IMPORTANT SECURITY/PRIVACY UPDATE

This update incorporates a fix for the Zcash protocol issue described in:
https://z.cash/support/security/announcements/security-announcement-2019-09-24/

In addition, this version also includes updates to Sapling functionality, including the Sprout to Sapling migration APIs added to the Zcash protocol up to and including version 2.0.7-3 of Zcash. The issue fixed by this update does not expose any funds to theft or counterfeit, but it does fix a vulnerability that an attacker may exploit to probe for information regarding a specific node's

control over private keys. Since the issue resolved does not put funds directly at risk, this update is considered CRITICAL, not mandatory. We recommended that all users upgrade to this version of the Verus CLI or GUI wallet immediately and discontinue use of previous wallets.

Notable Changes

- Integrated Zcash updates from Sapling release to 2.0.7-3
- Added Sprout to Sapling upgrade API
- Including support for PBaaS multi-chain test network with permissionless blockchain launches, reserve currency conversion protocols, merge mining, and easy cross-chain transactions.

Testnet Reset

The testnet will be deleted and relaunched for this release. **IF YOU HAVE PREVIOUSLY LAUNCHED VRSCTEST, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**
Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
%APPDATA%\Komodo\VRSCTEST
%APPDATA%\VerusTest
```

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol

.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE....

Assets 8

PBaaS Cross-chain Technology Preview September 23, 2019

Asherda
v0.6.0.34-PBaaS-Cross-chain-Technology-Preview
`b2c0e54`
PBaaS Cross-chain Technology Preview September 23, 2019 Pre-release

Notable Changes

- Properly distribute initial contributions on non-reserve chains
- Show chain name on failure to notarize

- Continue attempting to notarize after running out of notary rewards
- Fix getdefinedchains currency state

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol

.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
```

```
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE", "paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE", "paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Assets 8

v0.5.8

Asherda
v0.5.8
`1a1d05a`
v0.5.8

This release includes the ability to connect to and use the Verus Testnet with easy multi-chain network, cross-chain send, and in-wallet currency conversion capabilities that go beyond any blockchain system or cryptocurrency available today. Don't take our word for it. Just give it a try when you're not using the other mainstream wallet features

Try out cross-chain sends, create and exchange your own reserve currencies and merge mine up to 15 chains at once on the VerusCoin testnet!

Notable Changes

- Added Testnet support for Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

With no costs beyond public blockchain network fees, Verus PBaaS Reserve technology enables any organization of any size to make a fractional reserve currency of its own on an independent, uncluttered blockchain with cross-chain interoperability, and immediately useful for payments and conversion to other currencies from within the Verus wallet. 100% of the 0.01% fees for such conversions go directly to the miners and stakers of the organization's new chain, and those miners and stakers also have the option of merge mining up to 14 other blockchains and coins while mining their own project, improving efficiency, profitability, and shared liquidity for everyone involved. Each blockchain is independent and connected through the Verus Reserve protocol

.

Launching the testnet:

```
./verus -chain=VRSCTEST
```

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/462a2014bdd120eb155b964fe1f22910671ade38e26ff4372e64341e9613dff5/detection
Linux:
https://www.virustotal.com/#/file/6b1b3b848ab12c471d0469db37e5f8059aee565550d9e1c38fd97a55d493f184/detection
Windows:
https://www.virustotal.com/#/file/878ee7b0ed79d20f03994cf6322345be4a9f857aa4326ceb564d4ec51edd53ef/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:

https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview September 14 2019

Asherda
v0.6.0.33-PBaaS-Cross-chain-Technology-Preview
`ab04c6c`
Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview September 14 2019 Pre-release

Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Notable Changes

- Minor fix to export transactions

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Assets 8

Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview September 13 2019

Asherda
v0.6.0.29-PBaaS-Cross-chain-Technology-Preview
2accbf4
Public Blockchains as a Service (PBaaS) and Verus Reserve cross-chain currency conversion technology -- Technology Preview September 13 2019 Pre-release

Please note that this is a test release intended to use for testing and development with public blockchains as a service (PBaaS). THIS HAS NOT BEEN TESTED FOR USE ON MAINNET.

The Verus public testnet enables anyone to make new blockchains and currencies, each with scale and operational independence, zk-SNARK privacy, automatic wallet support, and 100% liquid conversion between VRSCTEST currency at any volume. Sending currencies cross-chain or converting between currencies is as easy as sending a cryptocurrency in less advanced systems today, except that the new Verus protocol enables you to include another blockchain as part of your destination and convert a fractional reserve currency to and from its reserve, according to its exact price along the way. All conversions are fully decentralized, require a standard protocol fee of 0.01%, which goes 100% to miners and stakers of the network, and calculates one price per block for all buys and sells in that block with zero spread.

Verus PBaaS Reserve technology enables any organization of any size to make a currency with its own blockchain, support it in world class wallets, develop its own applications, and use it for payments worldwide that can be converted to the reserve currency without ever having to leave the blockchain network. Each blockchain is independent and connected through the Verus Reserve protocol.

Notable Changes

- Fixed daemon output export error message on VRSCTEST
- Fixed fees bouncing back from a reserve send to Verus and allow exports to continue if done incorrectly
- Include currency state in chain definitions for both getchaindefinition and getdefinedchains

Chain definition:

- Maximum of 3 eras
- Minimum 500 blocks worth of notarizations.
- Minimum of 0.01 VRSCTEST per block of notarization.
  VRSCTEST are needed in wallet to make a chain.

Sample chain definition:

```
./verusd -chain=VRSCTEST definechain
'{"name":"RESERVEWITHPREMINE","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9j
PM","premine":50000000000000,"initialcontribution":5000000000000,"conversion":10
0000000,"minpreconvert":45000000000000,"maxpreconvert":100000000000000,"launchfe
e":0,"billingperiod":14400,"notarizationreward":100000000000,"eras":
[{"reward":5000000000,"decay":0,"halving":0,"eraend":0,"eraoptions":1}],"nodes":
[{"networkaddress":"ipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMcy5
Nxkx9jPM"},
{"networkaddress":"2ndipaddress:port","paymentaddress":"R9LkTaMn1XpjeorE4RjM6pMc
y5Nxkx9jPM"}]}'
```

The example above defines a chain that emits a premine to the indicated address and also converts all contributions up to a maximum amount to the native chain currency at launch at a 100%, 1:1 conversion price. All contributions will enter the currency's reserve deposits, and the more contributions on this chain, the higher the starting reserve ratio will be. If the minimum contributions are not met, the chain will fail to start and all contributions can be spent back to the original contributors.

The initial price in reserve at chain launch will start at 1.0, but there will be some fees to be paid in the early blocks that will come in as reserve currency. All fees in reserve that are converted to a fractional reserve currency are effective purchases of the fractional reserve currency, and based on a reversible formula, a currency's price will rise or fall on conversion from or to reserves with a level of volatility that is inversely related to the reserve ratio. A currency with a reserve ratio of 1 will never change its price when converting to or from its reserve. A currency with a reserve of 0.1 or 10% will rise or fall fairly quickly when converted to or from, depending on the amount of conversion relative to the total currency reserves on deposit.

Once a chain is made, anyone running the local VRSCTEST chain can connect to any defined chain with the command:

```
./verusd -chain=SOMECOIN
```

On first connection, a local config file is created, and VRSCTEST will not need to run to connect. However, it will be needed to merge mine.

The presence of a correct config file for any chain, which are kept in .verustest/PBAAS/CHAINNAME, which is the home directory for a PBaaS chain, enables the chain to load as an independent chain.

Here are a few examples of cross-chain operations using the command line:

Participating in a blockchain launch and converting from reserve before the chain has started (refund address is optional and will default to paymentaddress):

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "preconvert": 1}'
```

Sending unconverted VRSCTEST to an address on the RESERVEWITHPREMINE chain for later conversion to/from the native coin.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending reserve to another chain and converting a coin from reserve to that chain's native currency at the current market price.

```
./verus -chain=VRSCTEST sendreserve '{"name": "RESERVEWITHPREMINE",
"paymentaddress": "R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "refundaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Sending reserve currency to an address on the current PBaaS reserve chain with no cross-chain operation, Reserve currency on a PBaaS chain is equivalent to and convertible between the native VRSCTEST currency, but may also be sent to and used on PBaaS reserve chains. It will arrive as reserve currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000}'
```

Sending native currency to an address on a PBaaS reserve chain and converting it to reserve currency at the same time at the current market price.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "toreserve": 1}'
```

Sending reserve currency to an address on a PBaaS reserve chain and converting it to native currency at the same time at the current price. It will arrive as native currency.

```
./verus -chain=RESERVEWITHPREMINE sendreserve '{"paymentaddress":
"R9LkTaMn1XpjeorE4RjM6pMcy5Nxkx9jPM", "amount":5000, "tonative": 1}'
```

Getting the current price in reserve and other currency statistics for the most recent block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
```

... a specific block.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate blocknum
```

... a range with a specific period between.

```
./verus -chain=RESERVEWITHPREMINE getcurrencystate
startblocknum,endblocknum,blockstostep
```

Refunding contributions made to a failed PbaaS chain launch.

```
verus -chain=VRSCTEST refundfailedlaunch CHAINNAME
```

Skip to content

- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

- Code
- Issues 21

- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Releases: VerusCoin/VerusCoin

Releases Tags

v0.6.0-7

Asherda
v0.6.0-7
2491918
v0.6.0-7

At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

Notable Changes

- Improved staking check

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address

mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/387f4933627c7ad774c19ede52aba9d2ca65b27dca49e6dc6ee5bc471cbebcb0/detection
Linux-AMD64:
https://www.virustotal.com/#/file/fc157e8826f9f547e8ff54eed6ccd1cf8c314104e90a46f39464a9484c9597fb/detection
Windows:
https://www.virustotal.com/#/file/ff10726405536584fa9be5c2989c8365aaa5eaa79bb5bd6cd603d9e2f979e8dc/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-6

Asherda
v0.6.0-6
`a682ef2`
v0.6.0-6

At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

This release fixes an issue that does not generally affect the Verus network, but can corrupt individual wallets in the case of a specific type of reorg occurrence that is relatively rare, but can happen. We STRONGLY RECOMMEND that all users upgrade to version 0.6.0-6 asap.

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs

and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c4415da8270a48944cd29507129085a677ea09ba3cf014f93393a9f6fe08391c/detection
Linux-AMD64:
https://www.virustotal.com/#/file/599b02ca94c9409f4167c15f7c5203b909f38c43693df702bb0dd92d7cdbe098/detection
Windows:
https://www.virustotal.com/#/file/75b8aee40a5394f6b99ab0ac815b8973cc475d83bedb6e24cb7e37996553bd7c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-5

Asherda
v0.6.0-5
`fd371c1`
v0.6.0-5

THIS UPDATE FIXES AN OVERLY AGGRESSIVE BLOCK REJECTION IN THE VerusID PROTOCOL, AND IS A MANDATORY UPDATE FOR THE UPGRADED VERUS MAINNET

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WAS UPDATED TO INCLUDE VerusID. THIS UPDATE IS THE LATEST UPDATE FOR THE UPGRADED VERUS MAINNET

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery

state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8d9a74da37c9b1be5f416c2f2f92f772b086da01770d8ed6a118778f87a7e568/detection
Linux-AMD64:
https://www.virustotal.com/#/file/27aa9d8d4a89dadf1a87928664eef7dc70ef92624d62596ee6e699b064f17daa/detection
Windows:
https://www.virustotal.com/#/file/e8f9b060d37e7028714d8c078705c8d6ceaef51b0d6f4c121cfbe40ab781b16f/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-4

Asherda
v0.6.0-4
`f7cabaa`
v0.6.0-4

Notable Changes

- Alleviate peer banning behavior
- Staking enhancements

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WAS UPDATED TO INCLUDE VerusID. THIS UPDATE IS THE LATEST UPDATE FOR THE UPGRADED VERUS MAINNET

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of

UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a3c9e986b441a4973c15a3578fb713578b294b19077feedb2182a78270560d52/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3075c0ebc2337e36ce4d815a6a0d2a2715dcea2118cf7b70822061c60264c7e3/detection
Windows:
https://www.virustotal.com/#/file/594cf99ecc390e580edfbec4f7c73f45c959515030d256afb16e07074ba142e4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-3

Asherda
v0.6.0-3
`f0eb427`
v0.6.0-3

Notable Changes

- This update fixes an issue which would ban nodes too aggressively, resulting in disconnection from the chain and potential to fork away from the main chain. This should be considered a mandatory update.
- Add cansign and canspend to getidentity

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

**AT BLOCK 800200, THE VERUS MAINNET PROTOCOL HARDFORKED. THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET.

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/a82b8cfb4207b6a8e05300bc91c0363dbad0de7e3f5a1a57043b641ba4852b2e/detection

Linux-AMD64:
https://www.virustotal.com/#/file/1246d439115b7578ddd47f8427a93f3ebad9a637d0735ed1db31003a9a3f53be/detection
Windows:
https://www.virustotal.com/#/file/2ca4b8bef8ad3e23bb896593be1509cf85619f2a6b87e72f9e59b99b52b63810/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-2

Asherda
v0.6.0-2
`a680284`
v0.6.0-2

Notable Changes

- Enable optional VRSC or VRSCTEST chains as parent of an ID. For example, bob@ is equivalent to `bob.vrsc@` or, on testnet, `bob.vrsctest@`
- Update stratum protocol for mining to include solution information for miner, which enables smooth autoswitch and prepares for PBaaS merge mining
- Minor issues, including listidentities issue on Mac and compatibility with pool mining, even on block numbers under 16 to prepare for PBaaS

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which

will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/8a7f7bf3df09fa352979b1b92b3eea99fc802bce41f4349cfdeb3f25792233ce/detection
Linux-AMD64:
https://www.virustotal.com/#/file/de2961c3d4c0a00ce565989721d25b3abcf1e9e70b511c5cdb4e343005b13cc4/detection
Windows:
https://www.virustotal.com/#/file/74bbe68954498821f079a9a611cf6b8b8f8deb10f9a6e2feb594deba5c5ed7b1/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-1

Asherda
v0.6.0-1
`fdc64a5`

v0.6.0-1

Thanks to @hellcatz and his constant development, improvement, and support of LuckPool.net for Verus and other coins they mine, we have fixed an issue that would have allowed the old hash algorithm to be used after the VerusID and VerusHash 2.1 network activation.

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.

- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/021e59635bb77ccb8de45b017f5e07b1d71ab55e8f508e3a68aca96f7c9c2f5b/detection

Linux-AMD64:
https://www.virustotal.com/#/file/2c39e4fc74ba9a1874d5d3b9d957543c83ecca50f80f0be8f844b6a599a8b65c/detection
Windows:
https://www.virustotal.com/#/file/9f6a11ab3621496a0210d531167351a66c7b7881dfb405872b82db73c9094efe/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0 Verus ID - Decentralized, Friendly, Quantum-ready, Revocable, Recoverable Blockchain Identities for the World!

Asherda
v0.6.0
`be0d1c6`
v0.6.0 Verus ID - Decentralized, Friendly, Quantum-ready, Revocable, Recoverable Blockchain Identities for the World!

This release includes activation support for VerusID, and relative adjustment to the CPU and FPGA-equalizing VerusHash 2.1 Algorithm.

AT BLOCK 800200, THE VERUS MAINNET PROTOCOL WILL HARDFORK, AND THIS UPDATE IS MANDATORY TO REMAIN ON THE VERUS MAINNET. MAKE SURE TO UPGRADE BY DECEMBER 15TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it

made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction,

including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/bfa8df164e7ac5a1e7ea10e7abec275059df247fd9d62258e32b28777c40baf4/detection
Linux-AMD64:
https://www.virustotal.com/#/file/644b4e30c681f2268f18e9e258c696a8ab4ae53964bca4b0a2d256e407b464ec/detection
Windows:
https://www.virustotal.com/#/file/f9ad88116f33a1cc27ff75a4eb6108873a4ab3820759d0998b7a3a0e69d4d9b3/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.5.9-42 Verus Identity Release Candidate 3

Asherda
v0.5.9-42
8c1a6f6
v0.5.9-42 Verus Identity Release Candidate 3 Pre-release

Notable Changes

- Set content hashes will be a key/value map instead of an array
- Improve invalid block and invalid transaction handling.

Verus ID and VerusHash 2.1 Network Upgrade

This is the third release candidate (RC3) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.

This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today's centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate.

This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU.

VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.
2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:
    - **Primary -** this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.
    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.
    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.
3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.
4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@"`.
5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer’s ID
- 20 VRSC to the referrer’s referrer’s ID
- 20 VRSC to the referrer’s referrer’s referrer’s ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
That will respond with something like:
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your w...

Assets 8

v0.5.9-38 Verus Identity Release Candidate 2

Asherda
v0.5.9-38
f7326ce
v0.5.9-38 Verus Identity Release Candidate 2 Pre-release

Notable Changes

- Improved Identity synchronization
- Cleaned up debug log output
- Set block space limit for Identities

Verus ID and VerusHash 2.1 Network Upgrade

This is the second release candidate (RC2) for the next major upgrade to the Verus network. It is considered fully ready for testing on both VRSCTEST testnet and VRSC mainnet. While this release is likely to work fine on mainnet through the coming activation, it is not yet an official release that you should expect to support mainnet activation. If you install this version, please make sure to upgrade to an official release before network activation, which will happen at block 800200, expected to be mined or staked on December 15th.
This release enables two technology upgrades, one that would be considered big news to most cryptocurrency projects, an improvement to the FPGA-equalizing hash algorithm, and a revolutionary, new decentralized identity technology, Verus IDs, that will disrupt today’s centralized systems with the most secure, quantum ready, fully decentralized, self-sovereign digital identity system in the world.

Verus ID

Verus ID includes built-in privacy at the core through integration of zk-SNARKs, and provides a revolutionary improvement to blockchain address security, making it possible to actually recover your money after losing your keys, secure against identity and key theft, prove things about yourself without having to show more details than needed, and transfer assets to heirs as part of your estate. This release also introduces a new smart transaction technology that is unique to the Verus network and replaces its use of Komodo compatible crypto-conditions. The Verus ID system was made possible through the use of Verus Smart Transactions, which will be available for everyone to use on their own blockchains in the upcoming PBaaS network upgrade. Verus Smart Transactions use standard Bitcoin style serialization rather than the ASN.1 used in crypto conditions, which makes support on lite or mobile wallets simpler to implement than the crypto-condition protocol, which is still used on the Verus network for Stake Guard.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Verus ID - A Better Blockchain Identity Technology

If you've been around crypto for even a little while, you'll start to hear things like "not your keys, not your coins", "lose your keys, lose your coins". If you lose your keys, no one will ever be able to recover your money.

In fact, trading off decentralized, censorship resistant public systems for this risk to your cryptocurrency assets is such a fundamental issue in blockchain systems today that it is considered an acceptable price for having the benefit of full control over your own funds and blockchain assets.

Verus ID provides an elegant solution to these previously unsolved, fundamental problems, and also provides quantum-ready friendly identity names and blockchain funds addresses as self-sovereign, revocable, recoverable identities that provide the following advantages over today's blockchain systems:

1. Identity and currency address are one and the same. Your friendly name can be used as both a reference to who you are and an on-chain destination for any funds transfer.

2. Each identity must have a set of addresses (1 or more) and a minimum number of signatures required from those addresses to spend or sign on behalf of an identity. Each identity also refers to two additional identities that have authority over:
    - **Primary** - this is the self identity and may modify any part of the identity except its name, its parent, which is derived from the blockchain on which it is defined, and unless it is also one of the two other authorities, it may not modify any data under their control, including the identity of the other authorities once specified.
    - **Revocation** - this identity may revoke the primary identity, which will also revoke access to all funds and transactions under its control, but it may not spend any funds or sign on behalf of the identity.
    - **Recovery** - this authority may redefine/recover a revoked identity, but it may not change the revocation authority, nor may it modify an unrevoked identity at all.
3. Identities are transferable by defining them with primary addresses with private keys, which are under another party's control. When identities are transferred, the revocation and recovery authorities may or may not be modified as well, as long as the party updating the ID has the authority to do so.
4. If an identity is transferred from one party to another, all funds in transactions that are under the control of that identity will also transfer to the party to which control is being transferred. In the native wallet, these changes are automatically reflected in the wallet balance and visibility of the balance of an identity using `z_getbalance identity@”`.
5. If later, a new address type or signature algorithm, such as one that is quantum secure s added to the Verus blockchain, all transactions sent to an identity that is updated to be under the control of the new address type will be subject to its spend conditions as well. That means that if you have 1000 transactions sent to your identity name/address, an upgrade to a quantum secure address, when they are available, will retroactively secure all 1000 transactions under the control of that address.

Making a new identity

Arguably, the hardest part of the new Verus identity technology is just making your first identity, especially without a GUI. Once you have your own identity, a lot becomes easier, not the least of which is the ability to send money to a friendly name instead of a base58 blockchain address or hash. For example, once you have your identity, either of the following commands will work as well as just about any other CLI command that takes an address:

```
./verus -chain=VRSCTEST z_sendmany bob@ ‘[{“address”:”alice@”, “amount”:100}]’
```

or

```
./verus -chain=VRSCTEST sendtoaddress alice@ 100
```

Making a Verus ID requires two steps, steps that are designed to prevent potentially selfish miners from replacing your purchase of an ID from the Verus blockchain with their own purchase of the same name instead. To prevent this name “front running”, you must first commit to a name that you wish to have in your identity as its friendly name. Friendly names are unique to a blockchain, so if someone already has a specific name you must have, you will need to either be the first to purchase that name from the blockchain or acquire it from the controlling individual or organization.

Creating a name commitment registers your commitment, but keeps the name completely secret until you register it and reveal your commitment. When you register a name, you must pay 100 VRSC or 100 VRSCTEST, unless you have a referral from someone who already has an identity. With a referral, the price is 80 VRSC or 80 VRSCTEST. When you register an identity with the CLI wallet, the correct amount of money, 100 VRSC without and 80 VRSC with a referral is automatically spent along with your registration. The funds for a referral are distributed as follows:

- 20 VRSC to the referrer's ID
- 20 VRSC to the referrer's referrer's ID
- 20 VRSC to the referrer's referrer's referrer's ID
- 20 VRSC to the miner of the block in which the ID is mined

100% of all funds that are not paid to referrers go to miners or stakers of the blockchain.

To create an identity:

Create your commitment with the following command:

```
./verus -chain=VRSCTEST registernamecommitment Name youraddressorid
referrerfriendlyname@
That will respond with something like:
{
  "txid": "377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
  "namereservation": {
    "name": "Name",
    "salt": "7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
    "referral": "iSJLKp1hvn51Zg2Y6FBKjzLs9AAy7fomWN",
    "parent": "",
    "nameid": "iDXx9FPrAS5k2XCGss6FFmDQQMsts63uUg"
  }
}
```

You then wait for the commitment to be mined into a block, then use the information returned above as follows:

```
./verus -chain=VRSCTEST registeridentity
'{"txid":"377576f8ada1acc2aeb013ddf7a9ad86756cb990d4c5fc70b5a9a0fca43d727e",
"namereservation": {
    "name": "Name", "salt":
"7b981f0a1ff2593167ef078150d0116335a7f329f9403b0abaeca30a42d8876a",
"referral":"referrerfriendlyname@"}, "identity":{"name":"Name",
"primaryaddresses":["RKo5u8N1sStZu81fU8kaxhoDcFcJmNEwSp"],
"minimumsignatures":1,
"privateaddress":"zs14y0aa096em2as6n4fauyumqeywz45rfpze38c39fksgtjsac9vmms7fmstc
ylpaalm7rseu8838"}}'
```

As long as you have enough funds and you are registering a new identity on the blockchain, you will be able to use your new identity in place of an address as soon as the transaction created by the registration command is mined into the blockchain.
You may list the identities in your wallet with the c...

Assets 8

Skip to content

-
-

Releases: VerusCoin/VerusCoin

Releases Tags

v0.6.5-2

Asherda
v0.6.5-2
`ea97d52`
v0.6.5-2

This is a mainnet release with fix to prevent any recurrence of the recent minor fork at block 1002417 and integrated support for the new PBaaS testnet. This release is approved for use on mainnet nodes, and it is being released on Github prior to full GUI wallet release v0.6.5-2 to enable exchanges, pools, and infrastructure nodes to update immediately, as needed.

Notable Changes:

- Added `getsaplingtree` API
- Added support for Verus Testnet with PBaaS Multi-currency Tokens
- Resolved multi-block reorg issue on mainnet that can prevent successful reorg in rare, otherwise valid cases

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Verus PBaaS Multi-currency Tokens (TestNet only)

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the

block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/9221506334ae6562d31f441202a5cbc019e6790ebec06c61ddaedebd9425db92/detection
Linux-AMD64:
https://www.virustotal.com/#/file/2e03e9442d34fe118cbeb586a5cff0567e98cdb4c6bda7cedf110f7456f7c62c/detection

Linux-ARM64:
https://www.virustotal.com/#/file/ac5d54de262512ff02405da2d6182fe4741d3929d8218a333024e2649e42123c/detection
Windows:
https://www.virustotal.com/#/file/f5223182d7dc042bf0cfc383586d34327d3cd3c0955fa0db92812d41146f2298/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted text file.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AfeNDwABQR+hVp1D3jZymlkW4NrwdJbsh4jQDxxXf44WNf6QZoXY/UiU1WFy9RM+/pkCC1IQfCXF6I9e
y3B/3DofrvzKKAsV /Downloads/Verus-CLI-Linux-v0.6.5-2-amd64/Verus-CLI-Linux-
v0.6.5-2-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AfeNDwABQR+hVp1D3jZymlkW4NrwdJbsh4jQDxxXf44WNf6QZoXY/UiU1WFy9RM+/pkCC1IQfCXF6I9e
y3B/3DofrvzKKAsV
99d7451472bc1fb34359d17e524d7755defbd279b0c2569a67231f8b7a37f9ff
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.5-1

Asherda
v0.6.5-1
c3c4e12
v0.6.5-1 Pre-release

This release is intended for use in the VRSCTEST only. Mainnet approved testnet release is not yet available

Notable Changes:

- Check for creation of invalid staking transactions

Verus Testnet PBaaS Multi-currency Tokens

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn

the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/64f4f431c73cc8d28da48ee5adaa8fddb07ca20339706c1aa953abe8819cbbd4/detection
Linux-AMD64:

https://www.virustotal.com/#/file/bfa5cdbbf35495f9e6e4c8a71da7a05c41795c844aecf0d05817f7e5ee32f2aa/detection
Linux-ARM64:
https://www.virustotal.com/#/file/3c71f41cf03ea85fbc3d7c46b6fae18b9458bdd66d06f9dd16eea3a4dc0b3e05/detection
Windows:
https://www.virustotal.com/#/file/c671dff603e3a0219c9bd8403b8ac1efddb0485621dfce9b802e975b4d7652cc
/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Testing@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted textfile.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Testing@"
AcBbDwABQSAhpO83zww8L/K3dZTImWLcF8D/u93GyeQFCtKRoiZbb10cvNCMVDN/13S74mZUGCekQQV9
t4fzChTy1mJ/DKKk /Downloads/Verus-CLI-Linux-v0.6.5-amd64/Verus-CLI-Linux-v0.6.5-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Testing@"
AcBbDwABQSAhpO83zww8L/K3dZTImWLcF8D/u93GyeQFCtKRoiZbb10cvNCMVDN/13S74mZUGCekQQV9
t4fzChTy1mJ/DKKk
b589a80b79a022f24524e113ccd5b7841d137fb198c5e64618c7a3ed867feb62
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.5 Verus Testnet PBaaS Multi-currency Tokens

Asherda
v0.6.5
980d305

v0.6.5 Verus Testnet PBaaS Multi-currency Tokens Pre-release

This release is intended for use in the VRSCTEST only. Mainnet approved testnet release is not yet available

Verus Testnet PBaaS Multi-currency Tokens

The testnet supports a completely new capability of token definitions and token launches (Kickstarter/Gofundme style, ICO, ITO, IPO, etc.). These tokens can be sent through the Verus protocol using the new 'sendcurrency' API, which works for all currencies in the Verus network. Tokens can be used as reserve currencies for other currencies as well. Token launches can accept multiple other coins or tokens for conversion to the new token with payment to the token ID. Each token can have a different conversion price for pre-launch participation as well as minimums and maximums of participation. If minimums are not met by the currency's startblock, all participation will be refunded, less a minimal transfer/network fee, which was already allocated to miners/stakers.

Tokens can be controlled by the blockchain and used for many purposes, including payment models, tickets, point systems, etc., or they can be partially blockchain controlled, centralized tokens, with currencies that are mintable at any time by the ID behind them. The controlling ID can also receive receive aggregated outgoing transactions to external systems that both burn coins and pass account data in the process. This will allow applications to control currency supplies on the public network, whether these applications are external, decentralized networks or centralized custodial systems, in many typical use cases of tokens, including club coins, game tokens, token launches, etc., no programming is required.

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.

all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/f0cfd8bcd39b69e23481d265fb8789d27ae6e655ee168acb06573268ded74185/detection

Linux-AMD64:
https://www.virustotal.com/#/file/eb259e23a629cac31789621cf4e917905ac43ba722e47f82f8dc0360d4f6ba72/detection
Linux-ARM64:
https://www.virustotal.com/#/file/ea0b8244bb8c6ddea4eb0eac51e50aa9dfd57ebb0c17696f44c3e21b3d99320c/detection
Windows:
https://www.virustotal.com/#/file/80bc1d1c574a91708cbb4f13be9ba110cccc0c4fd8555bc4378737660eef6d06
/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted textfile.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AT9MDwABQR+giVSMbRiq3p60hCUVfEdm/yPLZYjX1Eh+ufvtnpDPhyjvjUGQRClss+mQohhXPxfWY5oq
IzfioqJsnHHlo2XH /Downloads/Verus-CLI-Linux-v0.6.5-amd64/Verus-CLI-Linux-v0.6.5-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AT9MDwABQR+giVSMbRiq3p60hCUVfEdm/yPLZYjX1Eh+ufvtnpDPhyjvjUGQRClss+mQohhXPxfWY5oq
IzfioqJsnHHlo2XH
60020bfebbb2ced560c5d1e1333c43d9f80c4db4797812213ae04ad39df99c22
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.4-3

Asherda
v0.6.4-3

`391c403`
v0.6.4-3

New in v0.6.4-3

- Added `z_viewtransaction` api: Gets detailed shielded information about an in-wallet transaction

Verus CLI v0.6.4-3

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which

will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c93d7f3ee35502a2dd9eaf87d76b253c6ee8e1b62c256103a267082daf3fa7f9/detection
Linux-AMD64:
https://www.virustotal.com/#/file/4346c377e8a6d95aae7abb289703a32ee1288786586a01fd0da6595c34d33ac9/detection
Linux-ARM64:
https://www.virustotal.com/#/file/131ec86c7a4024648d33c80992cfbd1708743fd4d1b9087234df6410d34ecb91/detection
Windows:
https://www.virustotal.com/#/file/6d57ec0805dc0f56604be718c83e61fe57fea9e640fcc3fb28e2bc8784619864/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted textfile.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
ATPPDgABQR/xBtY9wU+RmBJqX5Y6AWdPlw9nJShSAFpyGzu50ufFR3pTC1s1Wyk96zGTCZqMUzaurrbt
u4ysrKZNr7FUq/O/ /Downloads/Verus-CLI-Linux-v0.6.4-3-amd64/Verus-CLI-Linux-
v0.6.4-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
ATPPDgABQR/xBtY9wU+RmBJqX5Y6AWdPlw9nJShSAFpyGzu50ufFR3pTC1s1Wyk96zGTCZqMUzaurrbt
u4ysrKZNr7FUq/O/
3e5cc7bea46c6869545a0a886d66fdbf21fb2800761e3dca6f626046681ee26e
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.4-1

Asherda
v0.6.4-1
`ad586ce`
v0.6.4-1

New in v0.6.4-1

- Added `convertpassphrase` api, allowing wif key exports of electrum wallet passphrases

Verus CLI v0.6.4-1

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most

notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it's possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once

mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.

- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/21dff79a0060325787c96b4aaaf9becd161fe958cb4428adbc82d9bd1edb07cc/detection
Linux-AMD64:
https://www.virustotal.com/#/file/c4d339cc8a85b3beb240b3ea4692a7630124441305cf07cf7890724277df67f7/detection
Linux-ARM64:
https://www.virustotal.com/#/file/f8e92c5acf1fe5e0f317d4d0bbc8288c03b2ce4c21791a33ff5f2de60c385ca8/detection
Windows:
https://www.virustotal.com/#/file/d4c883ff9aa87e4e0356e66d44952a50e7bbff6fd8120a93b799510e20644a5a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AcFjDQABQSDvcPZH+EBJHnQR491hRjINjwGtaVf3S8wFwNAY0KwYF1Qj3z0kndcLIJKqDCVKWj5RarHj
CApwUGbjYK4x29Uc /Downloads/Verus-CLI-Linux-v0.6.4-amd64/Verus-CLI-Linux-v0.6.4-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AcFjDQABQSDvcPZH+EBJHnQR491hRjINjwGtaVf3S8wFwNAY0KwYF1Qj3z0kndcLIJKqDCVKWj5RarHj
CApwUGbjYK4x29Uc
0cbd162064918fb63a8914c0cf963e0eb310fcd100daa7433a9066c6806491e1
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.4

Asherda
v0.6.4
`31f26c9`
v0.6.4

New in v0.6.4

- Estimated active staking supply displayed in output from `getmininginfo`
- Average block fees for last 100 blocks displayed in `getmininginfo` output
- Immature coins, including time locked coins displayed in `coinsupply` output
- Eligible staking balance, with consideration for all disqualifying factors displayed in `getwalletinfo` output
- `z_exportwallet` as wallet backup now accepts boolean parameter to remove all addresses with no UTXOs or Identities in the backup
- Staking performance is improved
- Fixed rare condition of improper ID wallet synchronization of transferred or revoked IDs

Verus CLI v0.6.4

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed

already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/c90815f194641646e069824711fca2dc1f7a9cd6ba5bd5016bdcdcbff8a1f190/detection
Linux-AMD64:
https://www.virustotal.com/#/file/5ea61dee317688aee255f6f5255d62d0307517196098ace354881a49ee36a3c3/detection
Linux-ARM64:
https://www.virustotal.com/#/file/3e9e463afed0eaa737a6fbd51438d5e828fccd4ca3d4be74751e79fe3eecc119/detection
Windows:

https://www.virustotal.com/#/file/a3b8980fa5044177976e53e30bb3142c9ed78d982fede9d4cf015417dfa35f8a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AYpdDQABQR9QSLDca3cCXdpZhNtUyVL3GdFM4248ka0Nhpp0d+/yAGmH7pmBXfcyObnInU5cxGjgbe5n
2wLZRKmllS/zPM6q /Downloads/Verus-CLI-Linux-v0.6.4-amd64/Verus-CLI-Linux-v0.6.4-
amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AYpdDQABQR9QSLDca3cCXdpZhNtUyVL3GdFM4248ka0Nhpp0d+/yAGmH7pmBXfcyObnInU5cxGjgbe5n
2wLZRKmllS/zPM6q
60df4018e329cc6748379369d02943b1ff4e7b21363beedb2624ea2770fba1b9
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.6.2-1

Asherda
v0.6.2-1
beb316e
v0.6.2-1

Notable Changes

- Improved coinsupply API speed and added accurate immature funds count

Verus CLI v0.6.2-1

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you’d like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs

are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/4cf0e9684029faaa29b87277cbe19f85b40a059d6a04ae9861d94bebb6d95c56a/detection
Linux-AMD64:
https://www.virustotal.com/#/file/6bb4ebb2c4d5e0b59a52fb060d4b06596843955b3814dbf1c858a3e5d6be2ea9/detection
Windows:
https://www.virustotal.com/#/file/bd2fd47bb22dbe81bb34d74c04182d03e7e72f83a0cb254acb0b869d8ee07091/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AZPmDAABQR/rW5EvSr1wxFw4FP+FBgTbNoZJKROpreBeH8dGQx4YJFcjz40LDSBqWfufFFEFmaZGWNNW
Y3QVIp3jwU7qyLfu /Downloads/Verus-CLI-Linux-v0.6.2-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AZPmDAABQR/rW5EvSr1wxFw4FP+FBgTbNoZJKROpreBeH8dGQx4YJFcjz40LDSBqWfufFFEFmaZGWNNW
Y3QVIp3jwU7qyLfu
6bb4ebb2c4d5e0b59a52fb060d4b06596843955b3814dbf1c858a3e5d6be2ea9
```

The result will be true if the signature is valid.

```
true
```

Assets 8

v0.6.2

Asherda
v0.6.2
68b2bea
v0.6.2

Verus CLI v0.6.2

Worldwide Verifiable Digital Signatures for All!

Verus digital signatures, based on Verus ID, offer the first worldwide verifiable, decentralized, single or multi-sig friendly-name signatures for content such as messages and files, authentication, and attestations, with full support for revocation and recovery in case of key loss or theft.

Bitcoin, which was first to enable worldwide, P2P decentralized blockchain transactions, and most of its derivatives, offer ways to sign messages with specific private keys that can be verified against public addresses. While that does offer some limited signing capability, it does not offer capabilities that can compete with centralized services offering more sophisticated key management. A single public/private key pair lacks critical capabilities to make it suitable as an actual identity, most notable and obvious being the ability to recover from loss or theft of private keys, but arguably as important are friendly name aliases, modifiable multi-sig signing, updates, privacy, and the association of signed attestations by other identities to statements about properties of the identity.

Verus ID enables free verifiable digital signatures for all through the Verus ID protocol as just one of the many new capabilities it enables. It is also the foundation upon which many new applications and additional capabilities can be built.

For example, using Verus ID signatures, it’s possible for any journalist anywhere to sign photos, videos, and content, establish a reputation for authenticity, and counter the potential for deep-fakes to make the truth harder to find. Open source projects can now create their own identities and digitally sign their binary releases, ensuring that not only can a file be verified by hash as the one downloaded from a particular server, but by signature as the actual file initially signed by the developer or release engineer. Signatures also form the basis for any attestation of one party to the validity of another. In fact, there are so many applications for digital signatures, from things listed already, to physical entry systems, to workflow applications, to new earning opportunities that a full discussion of use cases would overwhelm these release notes.

In any case, we are happy to release digital signatures for all, and we hope you enjoy using this new, simple capability, maybe even think of a new use case you'd like to pursue yourself as a business opportunity on the Verus Network!

Verus ID

Verus IDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. Verus IDs provide plenty of opportunity for identity applications. Specifically, Verus ID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

Verus IDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.2 does not include quantum resistant signatures for transactions, Verus IDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. Verus IDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/32742e2b62aa446f6772ae68706626a953652c1766ac118cfb612ac6707bc48d/detection
Linux-AMD64:
https://www.virustotal.com/#/file/0eb57c65fc05daa34a55863f12987d4127cd8dd8c8fdce36df206e900598c9fe/detection
Windows:
https://www.virustotal.com/#/file/9aab61dc575c25e71df5c1c6d8c1eeb56d0b818e6a5d18378a412a7e929c810a/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation@".

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation@"
AfLZDAABQR9H2eDMg7vKCrf7KmJt/4+8Vi4FMJSrer93IUNCMhkw63JHnCXrIv1GysI3nhkY3qoCXGpW
QiIBlk6pzA/+Ztg8 /Downloads/Verus-CLI-Linux-v0.6.2-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation@"
AfLZDAABQR9H2eDMg7vKCrf7KmJt/4+8Vi4FMJSrer93IUNCMhkw63JHnCXrIv1GysI3nhkY3qoCXGpW
QiIBlk6pzA/+Ztg8
0eb57c65fc05daa34a55863f12987d4127cd8dd8c8fdce36df206e900598c9fe
```

The result will be true if the signature is valid.

```
true
```

Assets 8

v0.6.1

Asherda
v0.6.1
d0b3c87
v0.6.1

Notable Changes

- z_importviewingkey and z_exportviewingkey for Sapling to enable confidential, verifiable, and transparent voting

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which

will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/e1884e04a3d5932b99e3bfcb8a34cd3ed4034340367a5fade0b831eda6a0daa2/detection
Linux-AMD64:
https://www.virustotal.com/#/file/b514dbcf8ae804efe16ea5080751e3c0e6702a6db59b38eb7994c9a5fdc81c4c/detection
Windows:
https://www.virustotal.com/#/file/b5fb64cad0bb29ab76b0215b7050beee2919bcb1b6eef6bd52436d665fa3a051/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

v0.6.0-8

Asherda
v0.6.0-8
5dc785a

v0.6.0-8

At block 800200, The Verus mainnet protocol was updated to include VerusID. This update is the latest for the upgraded Verus mainnet.

Notable Changes

- Fix accretion of coinbases in mempool

Verus ID

VerusIDs are a fully functional blockchain protocol, not just an ID system. There is no corporation involved in the protocol, unlike most blockchain ID implementations. VerusIDs provide plenty of opportunity for identity applications. Specifically, VerusID provides:

Quantum-ready friendly crypto-addresses on the worldwide Verus network

VerusIDs can be used to receive and send funds, which are controlled by the single or multi-sig addresses specified in the identity itself. If these controlling addresses or the single or multi-sig properties are changed, which can be done by the controller of the identity, all future spends of UTXOs sent to that identity follow the updated spend conditions and are subject to the updated keys. Although Verus 0.6.0 does not include quantum resistant signatures for transactions, VerusIDs are themselves resistant to quantum attack with known algorithms, and we have already started to integrate a quantum secure signature scheme, which we expect to activate on mainnet early next year. When that is available, it will be possible to change an ID and have all of the funds sent to it made retroactively quantum resistant. Verus IDs can also be used to publish ID->destination address mappings on other blockchains, but only the Verus ecosystem has the ability to revoke, recover, inherit, funds in existing UTXOs.

Fully Decentralized Protocol

Anyone can create one and have complete, self sovereign control over it without permission to do so. All costs to create an ID go to miners, stakers, and ID referrers. VerusIDs are:

- **Revocable** -- each ID includes a revocation authority, which defaults to the identity self, and which has the permission to revoke the identity, which creates a valid transaction that, once mined into a block, prevents the identity from being used to spend or sign until it is recovered, effectively freezing all of its funds, for example, in the case of key theft.
- **Recoverable** -- each ID also includes a separate recovery authority, which also defaults to self, and which can recover the identity through redefining its primary state and the recovery state as well, though it cannot modify the revocation state, or vice versa, unless they are both controlled by the same underlying authority.
- **Private** - Each ID contains a set of zero-knowledge private addresses, which can be used as messaging, financial, or voting endpoints, and each ID also contains a content map of key-value hashes, intended to be used alongside applications and various identity policies to provide everything from private yet selectively provable claims and attestations to

selectively provable components of a strong passport, attested to with a quantum secure signature when that is available.

VerusHash 2.1

VerusHash 2.0 was the first algorithm to significantly equalize FPGAs dominance over CPUs, once they were introduced on the Verus network. While FPGAs were intentionally not blocked completely, which would simply drive the performance battle to the higher end and further into secret, the VerusHash 2.0 algorithm was developed to explicitly equalize FPGAs and modern CPUs and has met its original goals in keeping FPGA performance for the price under 2x of CPU. VerusHash 2.1 introduces an adjustment to the equalization technology, which we expect to tilt the balance a bit more favorably towards CPUs, while still enabling FPGAs to operate on the hash algorithm with minor modifications. Verus Developers have proactively reached out to FPGA manufacturers and made the new algorithm available to them, so that everyone will have an opportunity to mine and stake when the Verus economy starts to roll and identity rewards, which will not inflate the currency, but should far exceed the potential for block rewards, begin streaming from the network.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/6dc2eb44ee294829c6f14e38349ccd1f8ba2bf683b93c0cda328c9bbdd588bab/detection
Linux-AMD64:
https://www.virustotal.com/#/file/ba3014f6ae1298fea11258474eb053b8841b1be04db911e4a3e2f80990ac42c3/detection
Windows:

https://www.virustotal.com/#/file/596ef34389d9aa801936da73f39f7ac6c85878f9b927321537e850deae4f614f/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Assets 8

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-
-
-
-
-
-
-
-
-

Releases: VerusCoin/VerusCoin

Releases Tags

v0.7.1-5

Asherda
v0.7.1-5
8e50ee2
v0.7.1-5

New in v0.7.1-5

This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or later. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-5 has no improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-5 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocations":
[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do...

Assets 6

v0.7.1-4

Asherda

v0.7.1-4

67d1355

v0.7.1-4

New in v0.7.1-4

This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or later. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-4 has no improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-4 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do ...

Assets 6

v0.7.1-3

Asherda

v0.7.1-3

7e0042a

v0.7.1-3

New in v0.7.1-3

- This non-mandatory release is compatible with mainnet with mininal changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1 and 0.7.1-2, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1 or "-2". You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file.
- Improved staking supply calculation

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-3 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

Enhancements to TestNet Protocols for Increased Scale, Currency Burn Functions, and Centralized Fractional Currencies

Version v0.7.1-3 is not yet considered hardened for mainnet, but has increased scale over previous versions in addition to the ability to apply both fractional and centralized attributes to a currency. This allows price neutral minting by the currency controller through automatic reductions of the reserve ratio of all currencies. This version also enables a new parameter to the "sendcurrency" API called "burn". If "burn" is set to 1, a currency sent, which must be a token or fractional currency is officially removed from that currency's supply. In the case of a fractional currency, this will have the effect of increasing the on-chain price for all holders, depending on the amount burned.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are

simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:**

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to p...

Assets 6

v0.7.1-2

Asherda

v0.7.1-2

176ee5c

v0.7.1-2

New in v0.7.1-2

- This non-mandatory release is compatible with mainnet with no changes from v0.7.1-1. It contains fixes necessary to continue participating in the testnet launched in v0.7.1-1, without reset of your wallet or holdings, if you started using the VRSCTEST multi-currency testnet in v0.7.1-1. You should be able to synchronize easily, but if you are unable to synchronize VRSCTEST shortly after connecting, try deleting the data in your VRSCTEST folder, and retain the wallet.dat file

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-2 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts,

pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/1f9cb1fe4ba5aef4a0859aebd2b89001c72e1482fd27a99e1ee67f8de11011a7/detection
Linux-AMD64: https://www.virustotal.com/#/file/ccc03d8a038bf21af8...

Assets 6

v0.7.1-1

Asherda

v0.7.1-1
535988a
v0.7.1-1

New in v0.7.1-1

- Re-launched VRSCTEST with fixes. Make sure you delete your VRSCTEST data before connecting to the testnet.

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1-1 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts,

pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/e94319a3ad2b81825b0ce231deb1117ac28ae897ecee1dfb0fe424c198db7d06/detection
Linux-AMD64:
https://www.virustotal.com/#/file/a9eb0f6a0bec1feb8e156410e0ccdbb7139e16a4b10bdea5e939b30ad834f50f/detection
Linux-ARM64:
https://www.virustotal.com/#/file/f17a5af4d09123d3f11f3682929fdc92f46f474463134fbdc1c2da4e56bdf8f6/detection

Windows:
https://www.virustotal.com/#/file/aafa51f1d6068d65a80715db668e7e7e39b9d9437005de957d08f7300094a20c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or ...

Assets 6

v0.7.1

Asherda
v0.7.1
c35f5a7
v0.7.1

Verus Testnet with Multi-currency Fractional Reserves and Currency Launch Protocol

Verus multi-currency fractional reserve baskets are UTXO-based, fractional reserve currencies with anti-front running, simultaneously solved, provably fair liquidity system for all currencies and tokens in or imported into the PBaaS system.

While Verus daemon v0.7.1 has minor improvements for mainnet operation, it offers full support for the new Verus Testnet multi-currency technology preview. On testnet, the new "sendcurrency" API, which is already available on Verus mainnet in its simplified form, is enhanced with new multi-currency parameters. The first parameter, "currency":"currencyname" - where the default on testnet for currency name is "vrsctest", specifies the source currency of the send. The next, "convertto":"currencyname", allows conversion from one currency to either a fractional currency where it is a reserve or a reserve of the fractional currency of the source.

An end to front running

Conversion will always be at the same rate as all conversions processed in one group of transactions. There is also no spread between the conversion to and from a currency. All transactions in a aggregation of import blocks, which will be detailed in later documentation, are simultaneously solved in all currency conversion directions and all converted at the same rate, all getting the same price in each direction as any person who may be exchanging in the other.

There is a fee, 0.025% for conversions, 100% of which goes to miners and stakers. While it is not yet implemented, we also intend to add an implied volatility fee, which will be charged variably on imbalanced blocks of transactions, making blocks that change the price of a currency significantly pay more of an implied volatility fee, and ensuring that relatively volatile currency baskets offset any potential impermanent loss to liquidity providers risk with earnings from implied volatility fees. At the same time, currency baskets that are less volatile relative to the constituent currencies will typically have lower implied volatility fees and lower earnings for liquidity providers to offset a lower risk of impermanent loss.

The Best Way to Launch a Currency

Currency launches with Kickstarter-like minimums to activate or, if not met, automatically refund, dynamic currency launch pricing, based on participation, pre-launch participation price discounts, pre-conversion reserve currency carveouts, and price-neutral launch pre-allocations, all specified in easy to understand commands and parameters, no programming necessary!

Currencies can be launched to accept a range of other currencies and with or without fractional reserve capabilities. Verus import/export protocol was designed to make integration with other block chains provable and easy, and there are already external bridges in progress that expect to enable automatic send/receive of any Verus network token between Verus and ETH/ERC20 tokens, where currencies can be sent between Verus and ETH, expressed as Verus currencies on the Verus network, and exposed automatically on ETH as an ERC20 token.

Preparing for Multi-chain, Dynamic Merge Mining

While this 0.7.1 testnet enables on-chain token currency definition, the same advances will soon be available on the Verus merge mining and multi-chain technology which was running on testnet last year. When enabled, this will make it possible for Verus network chains to operate fully independently, yet be dynamically merge mineable along with Verus and up to 20 other Verus network blockchains on a single hash, earning rewards and powering all chains simultaneously. All network blockchains will provide the same fractional currency capabilities, both for on-chain tokens as well as native zk-SNARK supporting currencies as well.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):

```
verus -chain=VRSCTEST sendcurrency "bob@"
'[{"currency":"vrsctest","address":"alice@","amount":10}]'
```

Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"currency":"vrsctest","address":"alice@","amount":10},
{"currency":"VRSCTEST","address":"bob@","amount":10}]'
```

Converting VRSCTEST to another currency, TESTCOIN:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"TESTCOIN"}]'
```

Preconverting to new currency, NEWCOIN, before it is active:

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps it in a way that can be widely trusted to a specific, auditable store of gold.
I also would like to launch a token called mycoin@, which is something like a Kickstarter, where a business, "my", offers to attribute the coins some utility or product value if the purchase exceeds a certain level.
First, I could define the currency "gold" as follows:

```
./verus -chain=VRSCTEST definecurrency
'{"name":"gold","options":96,"currencies":["vrsctest"],"conversions":
[0.01],"minpreconversion":[1000],"preallocation":[{"mike@":50000000.00000000}]}'
```

of course, since this is a test currency, I send myself some to start. The identity of the currency must be funded with at least 10 VRSCTEST before sending the transaction returned from this command to
initiate a currency launch that will start at 50 blocks from when it was made (default), and that must have 1000 VRSCTEST preconverted at 0.01 VRSCTEST per GOLD in order to launch.
all of this happens as part of the mining process, since mining the blocks that launch a currency earn the 0.025% conversion fees of participation
in the launch, converting VRSCTEST to GOLD. I could send the following command before the block where GOLD token launches.
After it launches, the only way at present to create new tokens is with a centralized issuance option. To convert VRSCTEST to GOLD, you could issue the command:

```
./verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"mike@","convertto":"gold","preconvert":1,"amount":100}]'
```

that would effectively park my conversion until the token launches, at which point, I will either find 0.975 GOLD in my wallet, or I will have my VRSCTEST back.
Assuming it launches, and I later want to create mycoin, which can be converted to with either GOLD or VRSCTEST, I can create mycoin with:

```
./verus -chain=VRSCTEST definecurrency '{"name":"mycoin","options":96,
"proofprotocol":2,"currencies":["vrsctest", "gold"],"conversions":[0.01,
1.0],"minpreconversion":[1000,1000]}'
```

In "mycoin", I set proofprotocol to 2, which is PROOF_CHAINID. That means that the controller of the chain ID can mint new coins as follows:

```
./verus -chain=VRSCTEST sendcurrency "mycoin@"
'[{"address":"mike@","currency":"mycoin","mintnew":1,"amount":10000}]'
```

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE DELETE THE FOLLOWING**

DIRECTORIES BEFORE RUNNING THIS NEW UPDATE:

Linux:

```
~/.komodo/VRSCTEST
~/.verustest
```

MacOS

```
 Users/{YourUserName}/Library/Application Support/Komodo/VRSCTEST
 Users/{YourUserName}/Library/Application Support/VerusTest
```

Windows

```
"%APPDATA%"\Komodo\VRSCTEST
"%APPDATA%"\VerusTest
```

Launching the testnet:

```
./verusd -chain=VRSCTEST
```

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/7a46a8f579b45a08eaf9a20f138ff75c80590740959803c8c93a5f54810af3ad/detection
Linux-AMD64:
https://www.virustotal.com/#/file/26cb5bd36f7dcd4c8db49baaf427afea0aa87dc518dfac3bac93950d204944f9/detection
Linux-ARM64:
https://www.virustotal.com/#/file/2b07c7505f0c67d8957420677f85bb2a1947029494b01c2707fdb3d0c2644aaa/detection

Windows:
https://www.virustotal.com/#/file/1bf02b7b5022f3b4e9b072e604a6a94656a9dc9188d1625e557fe304bdf04067/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To fin...

Assets 6

v0.7.0-4

Asherda
v0.7.0-4
`ab82cc9`
v0.7.0-4

At block 1053660, the Verus Mainnet Protocol upgraded to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash that is now live on Verus mainnet.

The network upgrade activated the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new “sendcurrency” API for the command line provides more control when sending from and to multiple addresses or identities.
- “updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.
- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.
- Staking and hashing consensus protocol updates with VerusHash 2.2

New in v0.7.0-4

- Validate ID destination on `sendfrom`, `sendtoaddress`, `sendmany`, and `z_sendmany`

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/97fec5a44e617452d48e13f0c00736430d904a72d85fc61731fd01452de77880/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3c8134361c965ab87751f817a097c1825ca61e2a521bfd1ed9382fbc44638920/detection
Linux-ARM64:
https://www.virustotal.com/#/file/c5c70239f6cb94cd1ab8cb109597f609a198f1bce644fa3434f50426f9fa8724/detection
Windows:
https://www.virustotal.com/#/file/418d4da1a4113b29f7c7ce490f8aa527d6217f2d85fd10ff639b33110ab5eba0/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted text file.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AR1XEAABQR8XbEix6eohYN9m6BbLjcdUkCZH/NYw1/mfBq/mIT6SewPg4PvaJHsUIO8FLB8kQDpa9eKg
lWJN9DTfvpWlMH9Z /Downloads/Verus-CLI-Linux-v0.7.0-4-amd64/Verus-CLI-Linux-
v0.7.0-4-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AR1XEAABQR8XbEix6eohYN9m6BbLjcdUkCZH/NYw1/mfBq/mIT6SewPg4PvaJHsUIO8FLB8kQDpa9eKg
lWJN9DTfvpWlMH9Z
3c8134361c965ab87751f817a097c1825ca61e2a521bfd1ed9382fbc44638920
```

The result will be true if the signature is valid.

```
true
```

Assets 6

SECURITY UPDATE - v0.7.0-3 - MANDATORY

Asherda
v0.7.0-3
5611c0a
SECURITY UPDATE - v0.7.0-3 - MANDATORY

At block 1053660, the Verus Mainnet Protocol upgraded to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash that is now live on Verus mainnet.

Although the network is running smoothly after the upgrade, an issue was identified and resolved in this release. In addition, a couple of user features, including restoring "-pubkey=hexpubkey" as a parameter for redirecting staking and mining rewards are included.

The network upgrade activated the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new “sendcurrency” API for the command line provides more control when sending from and to multiple addresses or identities.
- “updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.

- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.
- Staking and hashing consensus protocol updates with VerusHash 2.2

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/08254d35e8693eb304b35d1cee55f03761b78ad76dc5e25d6e42b300fcdd4e50/detection
Linux-AMD64:
https://www.virustotal.com/#/file/aa4842153c54510626b171128265059da30ae2bbfca974ceecc4348abb67e544/detection
Linux-ARM64:
https://www.virustotal.com/#/file/8a23406c8726ccf2fced09a20f298590d334ea00dbbca02eeadfae3eb8349d95/detection
Windows:
https://www.virustotal.com/#/file/d274514424e2334a7695e74c20569e9267158f22303021539060da6d82ab482c/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted text file.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AfQaEAABQR9cFt4qNU1KD14hlvClLhfICrm9LWuzcyW8Wh+zsu/BOQji9pUp0PRBnEEH4aQWlfRgeS4j
2sByVDix3VMwj2aM /Downloads/Verus-CLI-Linux-v0.7.0-3-amd64/Verus-CLI-Linux-
v0.7.0-3-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AfQaEAABQR9cFt4qNU1KD14hlvClLhfICrm9LWuzcyW8Wh+zsu/BOQji9pUp0PRBnEEH4aQWlfRgeS4j
2sByVDix3VMwj2aM
d509506edcf48a3ba64d814916f98e46105a0f6b437635cf5d2f2247e1574d7a
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.7.0-1

Asherda
v0.7.0-1
9d4787b
v0.7.0-1

MAKE SURE TO UPGRADE BY June 14TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE.

At block 1053660, the Verus Mainnet Protocol will upgrade to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash will go live on the Verus mainnet as well.

The network upgrade will activate the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.

- A new “sendcurrency” API for the command line provides more control when sending from and to multiple addresses or identities.
- “updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.
- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.
- Staking and hashing consensus protocol updates with VerusHash 2.2

Notable Changes:

- Fix Komodo S4 notary count
- `sendcurrency` does not require coinbase funds to be shielded.

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/74ccab3336fc62efb8fb4cf1a37f3c50eba59030c906bf5475227e9276d29e4b/detection
Linux-AMD64:
https://www.virustotal.com/#/file/3cc1b3fd86368bb6a6c69f385e3411495974f4920dda96d9e8e6d48ef3aa0143/detection
Linux-ARM64:
https://www.virustotal.com/#/file/300379f4530e3a7f9379a55ffb3bb88ff7c1cb7806421bbfa8f3c1bb588bece7/detection
Windows:

https://www.virustotal.com/#/file/56dfa302af9a6185406c3db7102b75c77e67187d8d674b3c47374f8c28b90056/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted text file.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
AfcIEAABQSBXNoiaxzhPSEo99oIkVNEWvQP1ebMA5l4TNOlsUzi+DjoO9AcsQjAG9gF/
HuVJlS5gAgaQN28Pwj6ExYWoY/dm /Downloads/Verus-CLI-Linux-v0.7.0-1-amd64/Verus-
CLI-Linux-v0.7.0-1-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
AfcIEAABQSBXNoiaxzhPSEo99oIkVNEWvQP1ebMA5l4TNOlsUzi+DjoO9AcsQjAG9gF/
HuVJlS5gAgaQN28Pwj6ExYWoY/dm
0587c3da3d0f64015be98e097353a24d0581b9fbc1f3a788adf31a408c643023
```

The result will be true if the signature is valid.

```
true
```

Assets 6

v0.7.0 Verus Network Upgrade - MANDATORY UPDATE FOR MAINNET BY JUNE 14TH

Asherda
v0.7.0
`c1057a5`
v0.7.0 Verus Network Upgrade - MANDATORY UPDATE FOR MAINNET BY JUNE 14TH

MAKE SURE TO UPGRADE BY June 14TH, OR YOU MAY NEED TO RESYNCHRONIZE THE BLOCKCHAIN IF YOU ARE RUNNING IN NATIVE MODE.

At block 1053660, the Verus Mainnet Protocol will upgrade to the new Komodo Notary Nodes for Komodo dPoW notarization, season 4, and a number of new capabilities, as well as a change to the Verus Proof of Power staking and mining hash will go live on the Verus mainnet as well.

The network upgrade will activate the following features:

- Added support for season 4 Komodo notary node operators
- Coinbase shielding will no longer be required on any coinbases from the past that have not yet been shielded.
- Staking will now work on all normal ID balances, enabling full use of IDs for storing and staking funds.
- A new “sendcurrency” API for the command line provides more control when sending from and to multiple addresses or identities.
- “updateidentity` will now be able to change the case of characters in the name of a VerusID in the global locale.
- Revocation identity will now lose all control over an ID after revocation, including the ability to still change the revocation identity itself, which it used to retain. After revocation now, the recovery identity will have full control over the revoked ID.
- Staking and hashing consensus protocol updates with VerusHash 2.2

Disclaimer

This is experimental and unfinished software. Use at your own risk! No warranty for any kind of damage!
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The enclosed copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MacOS:
https://www.virustotal.com/#/file/67f38974426e598b60f3cc97ca4a0fd80f8291e24c6738ecea59ddf0055172e1/detection

Linux-AMD64:
https://www.virustotal.com/#/file/29919bf5117ec7aa88b86eae07e83d203909cd92a8cbb0a02911633e761cdff0/detection
Linux-ARM64:
https://www.virustotal.com/#/file/a9d5f4cff70ba5b7be4d236ae692affa846336a44f6ba1a398742d87fe3ec352/detection
Windows:
https://www.virustotal.com/#/file/7a7634d3bcc29b9a10190fd58a37be53ef686f7cb80293a0c4feb6aed9a056b4/detection

Avast and Kaspersky may flag the software as "not-a-virus" or "PUP". These are warnings that you are installing mining software, which may be installed by a third party to exploit your PC.
To find out more about the false positives, review the following resources:
https://blog.malwarebytes.com/detections/pup-optional-bitcoinminer/
https://www.kaspersky.com/blog/not-a-virus/18015/

Verifying Downloads

A txt file containing the signer, standard sha256 file checksum, and signature, is included for each download. These packages have been signed with the identity "Verus Coin Foundation Releases@".

1. Extract downloaded archive
2. Verify signature for the extracted archive using the extracted text file.
3. Extract archive to desired directory

CLI examples

Verifying file directly

```
./verus verifyfile "Verus Coin Foundation Releases@"
Afn5DwABQSBdcb9HEXsztsaPQLtxhwA1U31x7ahYamF+DqxIn3j+pk/WQEv79ewYd5pDgjp5AJ/
cpKWIfCVi468j1PbZ5WeJ /Downloads/Verus-CLI-Linux-v0.7.0-amd64/Verus-CLI-Linux-
v0.7.0-amd64.tar.gz
```

Verifying using a checksum

```
./verus verifyhash "Verus Coin Foundation Releases@"
Afn5DwABQSBdcb9HEXsztsaPQLtxhwA1U31x7ahYamF+DqxIn3j+pk/WQEv79ewYd5pDgjp5AJ/
cpKWIfCVi468j1PbZ5WeJ
f769349015e0f4b8a6da9b071e3342f2043bd6eec35a69a97de7c7b4fd910148
```

The result will be true if the signature is valid.

```
true
```

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-
-
-
-
-
-
-
-
    -
    -
    -
- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

Releases: VerusCoin/VerusCoin

Releases Tags

v0.7.3-3

Asherda
v0.7.3-3
`01e44d4`
v0.7.3-3

Mainnet Updates

v0.7.3-3 is a CRITICAL update with fixes for potential, targeted DOS attacks that do not affect overall network security, but should be considered mandatory for exchanges, pools, and businesses using the Verus network. Previous 0.7.3-x releases should no longer be used and should be replaced by this version.

While this release includes testnet functionality for Verus PBaaS, described below. This is a mainnet focused release that does not fix any issues on testnet. Testnet issues will be address in upcoming optional releases.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the

same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":t...
  ```

Assets 6

v0.7.2-8

Asherda
v0.7.2-8
d3cffbe
v0.7.2-8

Announcing MANDATORY CLI upgrade v0.7.2-8 - THIS UPDATE IS CONSIDERED MANDATORY - ALL NODE OPERATORS SHOULD UPDATE AS SOON AS POSSIBLE
This release includes the following improvements and updates:

- Critical updates
- Fixed fetch-bootstrap temp file cleanup on Windows.
- Fixed fetch-bootstrap checksum verification on MacOS

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a

list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.
- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.
- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I h...

Assets 6

v0.7.2-7

Asherda
v0.7.2-7
`3fde9bf`
v0.7.2-7

Announcing NON-MANDATORY CLI upgrade v0.7.2-7, with fixes necessary to continue participating in the current testnet. Mainnet features are effectively unchanged from v0.7.2-6.

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.
- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts,

protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide

time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one named gold@, one named mycoin@, and one named mike@. I would like to have one currency, gold@,
that I somehow launch in a way that maps i...

Assets 6

v0.7.2-6

Asherda
v0.7.2-6
`5b13e56`
v0.7.2-6

Announcing NON-MANDATORY CLI upgrade v0.7.2-6, with primarily testnet improvements. Mainnet features are effectively unchanged from v0.7.2-4, with only "z*" added as a new, supported wildcard to the `z_getbalance` API.
This release includes the following improvements and updates for testnet:

- Add `getcurrencybalance` API to get multi-currency balances on transparent/private addresses and identities.
- Optimize multi-currency operators and UTXO selection
- Display unlocked balance in `getwalletinfo`
- Add support for "z*" wildcard in `z_getbalance`

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it’s hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide

availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.
- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.
- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

Defining a Curren...

Assets 6

v0.7.2-5

Asherda
v0.7.2-5
1de9f7c

v0.7.2-5

Announcing NON-MANDATORY CLI upgrade v0.7.2-5, with fixes that only apply to testnet. Mainnet features are unchanged from v0.7.2-4.
This release includes the following improvements and updates:

- Fix testnet transaction funding issue for multi-currency `sendcurrency` commands
- Fix testnet currency conversions

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus

Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that

we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
"convertto":"BTC","via":"VRSC-BTC"}]'
```

- **Preconverting to new currency, NEWCOIN, before it is active:**

```
verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
"convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
```

Defining a Currency

To create a currency of a specific name, you need an ID of the same name. The controller of this ID is the only one who can create a currency of that name, and they can only do so once.
So, let's hypothetically assume I have 3 IDs, one...

Assets 6

v0.7.2-4

Asherda
v0.7.2-4
`bcd146e`
v0.7.2-4

v0.7.2-4 WITH THE WORLD'S FIRST FRIENDLY-NAME, zk-SNARK ADDRESSES FOR MAINNET AND TESTNET

Announcing NON-MANDATORY CLI upgrade v0.7.2-4 with the world's first friendly-name private z-address support for sending and receiving funds and messages, including the following improvements and updates:

- `sendcurrency` can now use sapling addresses as native currency funding sources and destinations for any transaction.
- An identity's z-address can now be referenced as a funding source or destination by appending `:private` to an ID (eg. `"idname@:private"`). This form of z-address will also be able to be used with the z_getbalance command.
- New wildcards can also be used as funds sources in the 'sendcurrency' command and parameters to z_getbalance that include:
    - All transparent addresses in the wallet (both R-addresses and IDs)
      `"*"`
    - All R-addresses in the wallet
      `"R*"`
    - All IDs controlled by the wallet
      `"i*"`
- `sendcurrency` will now be an asynchronous API that will return an operation ID, as `z_sendmany` does, and complete in the background. It will also have its status available through `z_getoperationstatus`.

NOTE: Due to the zero knowledge nature of z-addresses, a z-address is an endpoint in your VerusID which can be changed if you revoke and recover an ID, funds held on a private address or endpoint are not able to be redirected to the newly recovered ID unless you still have control over the z-address itself. Transparent funds controlled by an ID are recovered when the ID is revoked and recovered, even if you have lost the keys. This is a fundamental limitation of zero-knowledge privacy, not an error and not something to be fixed. Please consider this when using friendly-name private addresses as funds and not just message endpoints.

The Verus DeFi Testnet -- THERE IS NO TESTNET RESET

All new 'sendcurrency' features described above now work on both mainnet and testnet. In addition, testnet can source funds for DeFi features from private z-addresses as well.

This is the most groundbreaking testnet the Verus project has ever released, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.
- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus

Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that

we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

**Warning: All testnet coin...

Assets 6

v0.7.2-3

Asherda
v0.7.2-3
7d66ad8
v0.7.2-3

Announcing NON-MANDATORY upgrade v0.7.2-3 with the following improvements and fixes:

- Fixes a rare issue in wallet synchronization for IDs that have been transferred between wallets in a specific combination of transfers and spends of prior outputs. In the cases seen, the wallet maintained some outputs as unspent, when in fact they were spent on the blockchain. We have seen a very small number of people hit this issue after transferring IDs between owners and/or separate wallets and having spent prior outputs on transactions that still have valid UTXOs to the ID. If you have encountered such an issue, this version will

properly synchronize the wallet with a full rescan. If you have not, use of this version will prevent the issue from ever occurring.
- Stricter updateidentity paramater checks to ensure that revocation and recovery IDs are already registered, valid IDs on the blockchain before accepting a change

Testnet Reset

The testnet was deleted and relaunched on v0.7.2-2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2-2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.

- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts,

protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.

- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide

time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional curr...

Assets 6

v0.7.2-2

Asherda
v0.7.2-2
9a9d036
v0.7.2-2

Verus v0.7.2-2 is a mandatory for pools and notaries, but not generally mandatory Verus node and wallet upgrade, with support for the new Verus testnet (VRSCTEST) with fully decentralized finance, advanced ID capabilities, and DeFi liquidity system integrated into the new network protocol. While most of the changes in this release are focused on testnet, the mainnet support does include some improvements and a fix for the getblocktemplate API used by pools

Testnet Reset

The testnet was deleted and relaunched on v0.7.2-2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2-2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any

full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.
- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again

under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

Defining a Currency

To create a currency of a specific name, you need an...

Assets 6

v0.7.2-1

Asherda
v0.7.2-1
8b5a602
v0.7.2-1

This update is required to continue participating on the testnet launched in v0.7.2

Verus v0.7.2-1 is a non-mandatory but highly recommended Verus node and wallet upgrade, with fully decentralized finance and liquidity system activated on the new Verus testnet. While most of the new capabilities in this release are focused on testnet, the mainnet support does include some performance improvements and also fixes an issue that may occur in an edge case of revocation and recovery of IDs, which can result in a revoked ID being irrecoverable from an earlier version than v0.7.2.

Testnet Reset

The testnet was deleted and relaunched on v0.7.2. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO v0.7.2, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API, `getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.
- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.
- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to

enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
"timelock": <Unlock block delay>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":...
  ```

Verus v0.7.2 with Verus DeFi Platform and Theft-resistant VerusIDs on Testnet

Asherda
v0.7.2
`6b02f06`
Verus v0.7.2 with Verus DeFi Platform and Theft-resistant VerusIDs on Testnet

We are excited to announce Verus v0.7.2, non-mandatory but highly recommended Verus node and wallet upgrade, with fully decentralized finance and liquidity system activated on the new Verus testnet. While most of the new capabilities in this release are focused on testnet, the mainnet support does include some performance improvements and also fixes an issue that may occur in an edge case of revocation and recovery of IDs, which can result in a revoked ID being irrecoverable from an earlier version than v0.7.2.

Testnet Reset

The testnet was deleted and relaunched on this release. **IF YOU HAVE LAST LAUNCHED VRSCTEST FROM A VERSION PRIOR TO THIS, PLEASE FOLLOW THE INSTRUCTIONS AT THE BOTTOM OF THE RELEASE NOTES BEFORE RUNNING THIS UPDATE**

The New Verus DeFi Testnet

This is the most groundbreaking testnet the Verus project has ever announced, and we believe it holds the potential to improve and reshape the DeFi industry. There are truly so many new capabilities and blockchain firsts working now on testnet that it's hard to adequately cover them all. Below is a list of new capabilities active on the v0.7.2 testnet that will be on mainnet in the near future when considered fully hardened, tested, and ready for worldwide, public scale. Until then, Verus testnet is a fully decentralized blockchain network with 50% proof of work, 50% proof of stake, worldwide availability, and is running all of the new Verus protocols that are being tested and hardened for mainnet. All currencies and assets on the Verus testnet have absolutely no value and may be reset at any time in the interest of testing for mainnet. If you would like a representative basket of Verus testnet currencies that you can use to try out all of the new protocols from anywhere in the world, simply visit the Verus Discord and ask in the #pbaas-development channel. You can use this link as an invite to the Verus Discord: https://discord.gg/gecZsCf

- **DeFi and cross-currency liquidity integrated with blockchain processing, fees, VerusID, and Verus PoP, 51% hash-attack resistant consensus algorithm** - This is a massive set of platform capabilities that take DeFi further than any project in crypto today. We will list it here and describe it below. Using it is simple. Understanding the implications of what such a protocol can do when rent free and integrated in a truly decentralized public blockchain is a process of realizations that is likely to be similar to layers of an onion. We will do our best to explain below.
- **System-wide queries for fractional basket currencies for arbitrage, investment discovery, or conversion bargain hunting** - The new RPC API,

`getcurrencyconverters` (final name at mainnet, TBD), enables instant queries of any full node for all registered currencies above a fixed liquidity threshold that contain one or more reserve currencies. This new API enables wallets or applications to easily query the entire blockchain network of fractional currencies, determine which provides the best conversion rate at acceptable liquidity, and either convert at the best possible rates worldwide, or select only better-than-market rates and arbitrage to earn.

- **Enhanced VerusIDs with Verus Vault technology** - This capability extends the already groundbreaking VerusID revocable, recoverable, provable, friendly name blockchain addresses and identities to include an advanced time lock capability that does not prevent staking on locked funds, enabling easy implementation of things like vesting, trusts, protocols for theft-proofing funds, simplifying estate planning, and much more. With Verus Vault technology, your funds, identity, and estate planning can be done in a fully self-sovereign manner, with the peace of mind in knowing you are protected from lost or stolen private keys.
- **Verus Fee Pooling technology** A sustainable mining and staking fee and revenue model while strengthening, rather than creating risk of weakening blockchain security, by preserving immediate incentive to maximize per-block fees, while smoothing fee distribution to miners and stakers over time, fully addressing the legitimate concerns expressed and still not addressed by Vitalik Buterin with respect to ETH and ETH 2.0 (https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy/). In the fee pooling model, which is an extremely powerful, yet simple solution to enabling a mining and staking economy, all fees are gathered from transactions in a block as usual, then the fees, also calculated as usual, are added to a common, persistent pool of collected fees, and the current block miner or staker takes up to 1% of the aggregate fee pool.

Verus Vault - Advanced Time Locking of Identities

On the new Verus testnet, we have introduced a capability we cal Verus Vault, which both provides options for a theft-resistant workflow of identity management, but also enables many use cases from trusts to vesting schedules, to escrow alternatives, estate planning and more. The concept is simple, but the ramifications when combined with the already unique revocation and recovery capabilities offered by VerusID on mainnet are immense.

The Verus Vault use case, which was originally described in the conception of the feature, is the ability to timelock an ID, which would prevent any spending of any funds controlled by that ID, while still allowing the ID and all its funds to be used for staking. If the timelock was set for one day when locked, the unlock operation itself will require a waiting period of one day, which can only be circumvented by a revocation followed by recovery operation. In practice, this means that even if someone compromises your private keys while an ID is locked, they cannot spend any of the funds (though they could stake them and delegate rewards to another address, which you could immediately see) unless they unlock the ID first and wait one day. If you notice that your ID has been unlocked by someone other than you, you can revoke and recover before any funds are lost at all. If you wanted to get extremely secure, to the point of covering a case where someone may steal your private keys without you knowing and then lay in wait until you unlock your identity, so they can pounce on your funds before you lock it again, you can actually modify the controlling private

keys of an identity before unlocking it, which ensures that the unlocked funds become usable again under the control of freshly made, uncompromised keys. While we believe this unique Verus technology actually provides for fully theft-proof workflows and protection of all assets on the network under the control of an ID, we will generally refer to the technology as "theft-resistant" and disclaim all guarantees of fitness for any particular purpose as part of the "experimental software" disclaimer.

In addition to use cases related to theft resistance and delayed lock opening, Verus Vault can be used to provide more standard forms of time-locking by simply time locking and unlocking in one operation, leaving the unlock period immediately counting down. This capability could be used to provide time locked funds to team members, children, employees, or structured in a way to provide time locked deal and payment terms. The uses for this technology are certainly beyond those that we have already thought of, and we look forward to seeing how people leverage this powerful new protocol capability.

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all wallet funds to two recipients (alice@ and bob@):**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"currency":"vrsctest","address":"alice@","amount":10},
  {"currency":"VRSCTEST","address":"bob@","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via","VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

Defining a Currency...

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-

- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

- Code
- Issues 21
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Releases: VerusCoin/VerusCoin

Releases Tags

v0.8.0-1

Asherda
v0.8.0-1
`108ad28`
v0.8.0-1

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-1 UPDATE

This release includes a testnet fix for the Ethereum bridge and is required for those who want to continue using the current testnet.

v0.8.0 and v0.8.0-1 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.
2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.
3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.
4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.
5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.
6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.
7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can

now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular

liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the p...

Assets 6

v0.8.0

Asherda
v0.8.0

7274b43
v0.8.0

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY, RECOMMENDED v0.8.0 UPDATE.

This release is mandatory for those who want to use the testnet, which will be reset as of v0.8.0.

Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.
2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.
3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.
4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.
5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.
6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.
7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is ...

Assets 6

v0.7.4

Asherda
v0.7.4
77424d8
v0.7.4

ANNOUNCING HIGHLY RECOMMENDED NON-MANDATORY, CRITICAL FOR NOTARIES AND POOLS UPDATE, v0.7.4.

Version v0.7.4 can improve a node's synchronization and this version also upgrades and resets the Verus testnet, preparing it for the testnet launch of the Ethereum/Rinkeby bridge and bringing Verus PBaaS multi-chain, multi-currency, Verus Vault ID timelocking, Verus MEV-resistant DeFi, and the Ethereum bridge much closer to a mainnet ready state.

This release fixes all known functionality issues reported from the previous testnet including some related to currency and blockchain launches, as well as being incorrectly able to reduce the timelock period on a Verus Vault locked ID. It also includes the ability to export an ID from one chain to another, using the "exportid":true parameter in sendcurrency, enabling cross-chain use of VerusIDs. With this release, we will be resetting the Verus testnet. To use the new testnet, you must make sure to delete the vrsctest and .verustest/pbaas or VerusTest\pbaas folders beforehand.

In addition to being prepared for the Ethereum bridge launch, this release adds some new testnet capabilities, including the ability to send to IDs cross chain and export IDs from chain to chain using the `"exportid":true` parameter with `sendcurrency`. IDs only need to be exported once and can then be used and managed directly as a forked ID on the destination chain.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for

your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts

for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API re...

Assets 6

2 people reacted

v0.7.3-10

Asherda
v0.7.3-10
c3b2554

v0.7.3-10

Announcing NON-MANDATORY, RECOMMENDED UPDATE FOCUSED ON NETWORK CONNECTIVITY AND ROBUSTNESS IMPROVEMENTS v 0.7.3-10.

v0.7.3-10 has very minimal updates targeted at networking robustness - faster response to notary inconsistencies and a check for detecting a corrupt index that if present, might otherwise prevent a node from staying synced.

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

```
verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
"timelock": <Unlock block height>, "minimumsignatures": 1,
"primaryaddresses": ["t-address"]}'
```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob...
  ```

Assets 6

v0.7.3-9

Asherda

v0.7.3-9
`f86f251`
v0.7.3-9

Announcing immediate availability of NON-MANDATORY, RECOMMENDED ESPECIALLY FOR NOTARIES, HIGH PEER COUNT PUBLIC NODES, AND POOLS, v0.7.3-9 Verus CLI.

In addition to more immediate and effective bad node banning behavior, v0.7.3-9 includes a fix for signrawtransaction that could sometimes result in failure of countersigning a multisig, ID-based raw transaction.

While this release does include network robustness and a multisig fix, we were also able to upgrade Verus multisig capabilities to an ease of use and level of capability beyond other networks we are aware of today. In this release, it is now possible, on both mainnet and testnet, to issue any “sendcurrency” command with the source being a multisig ID, for which you do not have all the keys present in your wallet. If you do, there will be a return error in operation status, from both GUI or CLI, and in that error will be the partially signed transaction, which can be used by applications or with copy and paste to send to other signers with partial authority over the source ID. On mainnet, destinations for such a transaction can include IDs, private addresses with memos to a z-address or ID:private, and transparent addresses. On testnet, such transactions can include all that is possible on mainnet as well as multiple currencies, currency conversions, and cross-chain sends.

To our knowledge, this release raises the industry standard for multisig capability and its support on public blockchain protocols to a new level, even before considering the multisig revocation and recovery capabilities.

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts

for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that hav...

Assets 6

v0.7.3-8

Asherda
v0.7.3-8
`942b01b`
v0.7.3-8

Announcing MANDATORY UPDATE v0.7.3-8, which will re-enable full Verus Proof of Power, including staking, at block 1576200, occuring on Thursday at approximately 5:00PM UTC time. YOU MUST UPDATE BEFORE THEN TO FOLLOW THE MAIN CHAIN.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised

a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
  "exportto":"Bridge.PBaaSChain"}]'
  ```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -...
  ```

v0.7.3-7

Asherda
v0.7.3-7
2fcb26a
v0.7.3-7

Announcing MANDATORY UPGRADE v0.7.3-7. This release fixes the bug that was exploited by over the last day which stalled the network. You must upgrade to this release. If you shut down prior to block 15679999 you should be able to run this release without any additional steps. If your wallet continued to run after block 15679999 (or is still running) you will need to bootstrap or otherwise re-sync.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized "bridge" converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":...
  ```

v0.7.3-6

Asherda
v0.7.3-6
7a740d2
v0.7.3-6

Announcing MANDATORY UPGRADE v0.7.3-6, REQUIRED FOR KOMODO NOTARY SEASON 5. Notaries, pools, and exchanges should upgrade to this version as soon as possible, before notary changes take effect on block 1562500. Failing to update will not prevent connection to the network, but updating in a timely manner will ensure a stronger, notarized connection to the Verus blockchain.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized "bridge" converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"address":"RXLYm4J6qi7yam9zXtkEkRwbv...
  ```

v0.7.3-5

alexenglish
v0.7.3-5
`ef42f6e`
v0.7.3-5

Announcing NON-MANDATORY GUI upgrade v0.7.3-5, required to continue participating in the PBaaS enabled testnet. v0.7.3-5 resolves an ID wallet sync issue for mainnet that could occur when recovering a revoked ID, causing the wallet to double count some amounts. This issue could not result in either loss of funds nor a security issue. This release also resolves all known testnet issues on the current testnet and does not require a testnet reset.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized "bridge" converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee

conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.

- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

  ```
  ver...
  ```

v0.7.3-4

Asherda
v0.7.3-4
72c69c8
v0.7.3-4

Announcing NON-MANDATORY CLI upgrade v0.7.3-4, required for to continue participating in the PBaaS enabled testnet. Mainnet features are effectively unchanged from v0.7.3-3

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an

unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized "bridge" converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 2,
  "timelock": <Unlock block delay>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- Revoking an identity will clear its locked status, regardless of time delay or unlock height.
- A locked identity cannot revoke itself.

Conversion Queries

The `getcurrencyconverters` API retrieves all currencies that have at least 1000 VRSC in reserve, are greater than 10% VRSC reserve ratio, and have all listed currencies as reserves

- **E.g. BTC ETH:**

  ```
  verus -chain=VRSCTEST getcurrencyconverters btc eth
  ```

  Will return all currencies that have btc/eth markets at or above the liquidity threshold.

Sending and Converting Currency

Warning: All testnet coins/currencies have no value and will disappear whenever VRSCTEST is reset

The `sendcurrency` API can be used to send and convert funds.

- **Sending VRSCTEST from a single address (bob@) to a single recipient (alice@):**

  ```
  verus -chain=VRSCTEST sendcurrency "bob@"
  '[{"currency":"vrsctest","address":"alice@","amount":10}]'
  ```

- **Sending VRSCTEST from all private wallet funds to two recipients with friendly-name z-addresses (alice@:private and bob@:private):**

  ```
  verus -chain=VRSCTEST sendcurrency "*Z"
  '[{"currency":"vrsctest","address":"alice@:private","amount":10},
  {"currency":"VRSCTEST","address":"bob@:private","amount":10}]'
  ```

- **Converting VRSCTEST to a fractional basket currency, VRSC-BTC using IDs as a funding source:**

  ```
  verus -chain=VRSCTEST sendcurrency "*i" '[{"address":"bob@","amount":10,
  "convertto":"VRSC-BTC"}]'
  ```

- **Converting VRSCTEST to another reserve, BTC through a fractional currency, VRSC-BTC:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"bob@","amount":10,
  "convertto":"BTC","via":"VRSC-BTC"}]'
  ```

- **Preconverting to new currency, NEWCOIN, before it is active:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"alice@","amount":10,
  "convertto":"NEWCOIN", "preconvert":true, "refundto":"alice@"}]'
  ```

- **Sending VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -chain=VRSCTEST sendcurrency "*"
  '[{"address":"RXLYm4J6qi7yam9zXtkEkRwbvCrnWKGZuv","amount":10,
  "exportto":"Bridge.PBaaSChain"}]'
  ```

- **Converting VRSCTEST cross-chain to PBaaSChain:**

  ```
  verus -chain=VRSCTEST sendcurrency "*" '[{"address":"RXLYm4J6qi7...
  ```

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-
-
-
-
-
-
-
-
    -
    -
    -
- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

-
-
-
- Code
- Issues 21
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Releases: VerusCoin/VerusCoin

Releases Tags

v0.9.1-1

Asherda
v0.9.1-1
81dac44
v0.9.1-1

ANNOUNCING MAINNET AND PBAAS TESTNET UPGRADE v0.9.1-1

CRITICAL FOR NOTARIES, POOLS, EXCHANGES, MANDATORY FOR TESTNET USERS, HIGHLY RECOMMENDED FOR ALL MAINNET USERS

v0.9.1-1 includes testnet fixes for the Ethereum bridge launch with advanced features, addresses all known issues on testnet, and includes sync improvements for testnet and mainnet.

Testnet Featuring Verus PBaaS

The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.1, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Verus PBaaS Features Live on v0.9.1-1 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.
- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.
- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.
- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.
- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).
- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.

- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:
    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.
- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.
- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited...

Assets 6

v0.9.1

Asherda
v0.9.1
`623c69f`
v0.9.1

ANNOUNCING NON-MANDATORY, RECOMMENDED FOR MAINNET, MANDATORY FOR TESTNET PBAAS UPGRADE v0.9.1

Testnet Featuring Verus PBaaS -- The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

v0.9.1 improves mainnet bootstrap and GUI backup/restore user experience but does not upgrade mainnet protocols and is not a required update for mainnet-only users. It is required to use the public testnet. IF YOU HAVE ALREADY USED TESTNET WITH PRIOR VERSIONS, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Verus PBaaS Features Live on v0.9.1 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.
- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.
- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.
- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.
- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).
- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.
- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when

they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:
    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.
- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.
- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-

knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The...

Assets 6

v0.9.0-3

Asherda
v0.9.0-3
a2c4774
v0.9.0-3

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-3. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is recommended for testnet users to ensure the best experience with testnet.

This version fixes the following issues with v0.9.0-2

- Adds GUI support for seeing, making, and taking offers for currency or identity trades on the VerusID tab and when searching for identities.
- Fixes an issue where offers to buy or sell IDs were prematurely closed, even when expiry was set far in the future
- Fixes an issue with setidentitytimelock that caused it to return an error in most cases
- Ensures full rescanning of identities when using z_importwallet

New Command and API for Verus Vault - setidentitytimelock

v0.9.0-2 introduced a new command and RPC API, "setidentitytimelock". setidentitytimelock may be used to set a time unlock delay, set a lock until a specific block height, or unlock a delayed timelock on a VerusID under your control.

You can use setidentitytimelock in one of two ways:

```
verus setidentitytimelock '{"unlockatblock":1796500}' (returntx)
```

or

```
verus setidentitytimelock '{"setunlockdelay":1440}' (returntx)
```

or

```
verus setidentitytimelock '{"unlockatblock":0}' (returntx)
```

The first example will lock the ID until the block specified. If the block specified is in the past, the ID will not be locked.
The second example will set a 1440 block unlock delay, which will be needed to wait before spending funds controlled by the ID after unlocking the ID (about one day unlock time).
The third example will unlock an ID that has been locked with an unlock delay, and the unlock time will be set to the block height at which the ID was unlocked + the originally set unlock delay.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers...

Assets 6

v0.9.0-2

Asherda
v0.9.0-2
0c30a49
v0.9.0-2

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-2. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is mandatory for testnet users to ensure reliable connection to the current testnet.

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER. IF YOU HAVE NOT UPDATED AND ATTEMPTED TO STAY CONNECT TO MAINNET, YOU MAY NEED TO BOOTSTRAP OR RESYNC YOUR NATIVE WALLET OR NODE AFTER UPGRADING.

New Command and API for Verus Vault - setidentitytimelock

v0.9.0-2 introduces a new command and RPC API, "setidentitytimelock". setidentitytimelock may be used to set a time unlock delay, set a lock until a specific block height, or unlock a delayed timelock on a VerusID under your control.

You can use setidentitytimelock in one of two ways:

```
verus setidentitytimelock '{"unlockatblock":1796500}' (returntx)
```

or

```
verus setidentitytimelock '{"setunlockdelay":1440}' (returntx)
```

or

```
verus setidentitytimelock '{"unlockatblock":0}' (returntx)
```

The first example will lock the ID until the block specified. If the block specified is in the past, the ID will not be locked.
The second example will set a 1440 block unlock delay, which will be needed to wait before spending funds controlled by the ID after unlocking the ID (about one day unlock time).
The third example will unlock an ID that has been locked with an unlock delay, and the unlock time will be set to the block height at which the ID was unlocked + the originally set unlock delay.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new

age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidit...

Assets 6

v0.9.0-1

Asherda
v0.9.0-1
`d0aecf8`
v0.9.0-1

Announcing NON-MANDATORY, RECOMMENDED Verus v0.9.0-1. This release is considered non-mandatory but IS RECOMMENDED for mainnet users, Komodo notaries, and is mandatory for testnet users to ensure reliable connection to the current testnet.

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute

transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyoe with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, pr...

Assets 6

v0.9.0

Asherda
v0.9.0
4c7a0a6
v0.9.0

Announcing MANDATORY Verus v0.9.0 Mainnet Release of Verus Vault and the on-chain NFT and ID Marketplace

TO REMAIN CONNECTED TO THE MAINNET VERUS BLOCKCHAIN, YOU MUST UPGRADE FULL NODES TO v0.9.0 OR LATER BY BLOCK 1796400, which is expected to occur on or around Monday, November 29th at 6:15PM UTC.

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyoe with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your

funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-

capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-...

Assets 6

v0.8.0-5

Asherda
v0.8.0-5
4864da8
v0.8.0-5

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-5 UPDATE - ONLY REQUIRED FOR THOSE CONTINUING TO HELP GET TESTNET READY FOR MAINNET

We want to thank the Verus Community for bearing with the number of updates we've had to get the bridge to Ethereum working smoothly. This release both fixes the remaining known bridge issues and adds security to ensure that you don't send to an unspendable address on any network by using the wrong format for a destination.

The good news is that throughout the whole process of this first bridge release on testnet, we've been able to update through the issues we've seen without a reset and have not lost a single satoshi of test funds (which is not to say it is fully hardened quite yet). At this point, we hope and believe that this is the last update in this round of testnet, which will allow us to get back to focusing on the mainnet release and not asking those testing to update until we are very close to a mainnet release.

Once more, we will ask that you shut down the old version now, and we will restart the notarizations and transaction flow between Ethereum and Verus at 5 AM UTC time.

Thank you to everyone who helped with all the testing and exposing of issues that enabled us to make maybe even inconveniently quick progress.

If you miss the fork for this version at the above time you can get back on the correct chain by stopping Verus and deleting deleting everything other than your wallet.dat file in your data directory at:

- Linux: `~/.Komodo/vrsctest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest` or `%AppData%\Komodo\vrsctest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
```

```
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an

unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized "bridge" converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains.

Locking and Unlocking IDs

- **Time Lock:**
  The timelock parameter defines the unlock height of the identity.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags": 0,
  "timelock": <Unlock block height>, "minimumsignatures": 1,
  "primaryaddresses": ["t-address"]}'
  ```

- **Time Delay:**
  The timelock parameter defines how many blocks to delay an ID's unlock when the flags are set back to an unlocked state.

  ```
  verus -chain=VRSCTEST updateidentity '{"name": "ID@", "flags...
  ```

Assets 6

1 person reacted

v0.8.0-4

Asherda
v0.8.0-4
21d7af5
v0.8.0-4

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-4 UPDATE.

This release has no mainnet changes, resolves issues that people using testnet have reported with the Ethereum/Rinkeby bridge, and is mandatory for those who wish to continue using testnet after transactions between the Verus and Rinkeby testnets resume flowing.

If you have pending transactions on the bridge, some requested conversions may come back as refunded, and you should receive all amounts expected, less fees. We will not reset the testnet, but within an hour after this availability, the network will resume transaction imports/exports to and from Rinkeby, and anyone who continues running the old version may need to resync to testnet from scratch to continue. If you have upgraded to v0.8.0-4 or do so before connecting again to testnet, you should only see improvement without disruption.

I want to thank everyone who participated already in using this testnet and bridge release. I hope we can keep that level of participation or more up, as it has been extremely helpful. Because of all that's happened on this testnet to date, we were able to make a lot of progress in finding and resolving the following testnet issues in this release:

1. Enable support for much larger proofs than was previously possible from systems that either have large proofs required or significantly variable length proofs. Having this capability in the daemon will pave the way for more types of cross-chain proving in the future as well.
2. Fix failed refunds of conversion transactions attempted after a fractional currency launches, but before it's preconversion phase is complete. If you sent transactions during this time, you may have stuck refunds that should flow after this upgrade.
3. Ensure that currencies are marked as "launch complete" immediately after all preconversions have been processed, enabling normal conversions in all directions from that point forward.
4. Fix an issue that left some import notarizations incomplete and caused cross-chain transfers to wait longer than expected after some imports.
5. Improve and validate hardening.

If you miss the fork for this version at the above time you can get back on the correct chain by stopping Verus and deleting deleting everything other than your wallet.dat file in your data directory at:

- Linux: `~/.Komodo/vrsctest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest` or `%AppData%\Komodo\vrsctest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the ...

Assets 6

v0.8.0-3

Asherda
v0.8.0-3
`d6c4d90`
v0.8.0-3

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-3 UPDATE.

THIS UPDATE IS ONLY NEEDED FOR THOSE COMMUNITY MEMBERS USING TESTNET. THERE ARE NO MAINNET CHANGES IN THIS VERSION.

You will need this update to stay connected to testnet once we restart bridge traffic in about an hour. If you do stay connected on the old version and fork away from testnet, you should still be able to start with a reindex and rescan after upgrade.

v0.8.0 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.
2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.
3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.
4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.
5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.
6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.
7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and sta...

Assets 6

v0.8.0-2

Asherda
v0.8.0-2
292c723
v0.8.0-2

Announcing immediate availability of GUI and CLI for all platforms of the NON-MANDATORY v0.8.0-2 UPDATE

This daemon release includes a fix that simplifies some bridge transactions, and is the version you should run on testnet when the Ethereum bridge does launch for a good experience.

v0.8.0 Updates include:

1. Testnet reset with fixes for all reported PBaaS, DeFi, and advanced VerusID user issues. If you had reported issues, please verify that the issues you reported are addressed in this release.
2. Support for the bidirectional Ethereum gateway/bridge, which has been testing on private networks and which we hope to launch within a day or two after first vetting it on the release testing network with community test volunteers.
3. Support for the new `getvdxfid` RPC call, used along with your ID and published names to generate VDXF (Verus Data eXchange Format) keys, which community members used to create the world's first self-sovereign, completely decentralized, rent-free, non-cancelable social media profiles as can be seen here: https://luckpool.net/profile/identity/mike.vrsc . Work is underway to document the VDXF keys defined and used for profiles as well as the process of setting up your own. While the capability is already extremely powerful, we should remind everyone that Verus is a platform, not a social network. The core technology is exciting, but what will be even more exciting is when businesses and entrepreneurs leverage it to enable new, self-sovereign user experiences, applications, and accompanying businesses to enable the future, truly decentralized web.
4. Fix for the Electron certificate issue in the GUI wallet, which has been affecting prices and preventing BTC fee calculation.
5. Support for some additional, popular ERC20 currencies, in anticipation of more usage after release of the ETH bridge.
6. Fix in GUI for calculated balances of specific addresses sometimes showing lower than actual, even though wallet balance displayed correctly.
7. Additional hardening, fixes and improvement focused on mainnet and the Ethereum bridge.

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/VRSC`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\VRSC\`, `%AppData%\Roaming\VerusTest`

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and

fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with a...

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy

- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-
-
-
-
-
-
-
-
    -
    -
    -
- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

-
-

Releases: VerusCoin/VerusCoin

Releases Tags

v0.9.4-3

Asherda
v0.9.4-3
2521515
v0.9.4-3

Announcing v0.9.4-3, UNCHANGED FOR MAINNET -- MANDATORY FOR CONTINUED TESTNET USE

This release has no material mainnet changes.

This release upgrades the testnet protocol to fix an issue with the Ethereum bridge and VRSCTEST mining and staking. Please update to v0.9.4-3 to continue mining and staking on testnet and the Ethereum bridge. This version also fixes an issue discovered when defining Ethereum NFT mapped tokens, which are not yet supported in the released Ethereum contracts, but are updated no in the daemon and ready to use when the contracts are upgraded, hopefully in the next few days.

NFT mapped tokens will allow you to define an Ethereum NFT as the token that controls revoke and recover for an ID. Similar to how you can define a token for tokenized ID control, an NFT mapped token, when defined, refers by contract and tokenid to a specific NFT on the Ethereum blockchain. Once defined as the currency for a specific ID or sub-ID and linked to the Ethereum NFT, that NFT will be able to be sent over from the Ethereum blockchain as a satoshi of that currency, sent to any address or ID on the Verus blockchain, exchanged trustlessly on the Verus on-chain P2P marketplace, and used to revoke and recover the ID which defined the token. Once the contracts are updated, you will be able to mint NFTs on Ethereum's Goerli testnet, and endow your Ethereum NFTs with all of the power of your specially named VerusID. This new technology allows you to protect your NFTs with VerusID's revoke and recover technology, bind your NFT to the pseudonymous ID you use to log into applications, use it to extend and publish much more than just the NFT, and generally take your Ethereum NFTs to the next level!

The rest of these release notes are the same as the prior release. Drop by the #pbaas-development channel to help us prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
```

```
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an ...

Assets 6

v0.9.4-2

Asherda
v0.9.4-2
1541b4c
v0.9.4-2

Announcing v0.9.4-2, UNCHANGED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release has no material mainnet changes.

THE NEW ETHEREUM BRIDGE IS EXPECTED TO BE DEFINED WITHIN THE NEXT 24 HOURS AND FOR THE BRIDGE LIQUIDITY BASKET TO LAUNCH SHORTLY AFTER DEFINITION. IF YOU LEAVE A PRIOR VERSION NODE RUNNING UNTIL AFTER THE VETH BRIDGE IS DEFINED, YOU MAY FORK AWAY DUE TO CHANGES IN THIS RELEASE AND THEN NEED TO RESYNC TO TESTNET FROM SCRATCH TO CONTINUE.

For testnet, this is RC4 of the Verus PBaaS protocol, which includes some Ethereum bridge related fixes and a merge mining API change. With the API change, all merge mining nodes must upgrade to the new protocol to be compatible with each other and participate in cross-chain notarization rewards.

After the Ethereum bridge has been defined, you will be able to send currencies over to Verus testnet from Ethereum's Goerli network and also participate in the launch of the Ethereum bridge liquidity basket with VETH, USDC, or VRSCTEST by using `sendcurrency` with `"preconvert":true` on the command line or through the GUI. We recommend that you get some Goerli ETH using this facet or others (https://goerli-faucet.mudit.blog/), which will make your experience of all Verus PBaaS can offer a bit more well rounded when the Ethereum bridge is available once again.

Once the Goerli Ethereum liquidity basket (Bridge.vETH) activates on testnet, which is a 100% backed basket composed of 1/3rd each of Verus, Ethereum, and USDC (all testnet tokens), you can use it from Metamask on Goerli, from the Verus CLI or GUI on the vrsctest chain, or from CLI/GUI on other PBaaS chains to provide liquidity while holding Bridge.vETH, a 100% backed basket of three currencies of balanced relative value, or to pass through and convert currencies between Verus, Ethereum, USDC, and Bridge.vETH.

The Brige.vETH currency also coordinates with the vETH@ gateway and enables registering `*.veth@` IDs for $5 worth of Bridge.vETH, based on its internal USDC conversion price at the time of registration. When you register `*.veth@` IDs, 0.02 VRSCTEST goes to miners and stakers, and $5 worth of Bridge.vETH is burned, reducing the total supply of Bridge.vETH for everyone without reducing reserves, and contributing to the relative value, "real yield" of Bridge.vETH, relative to its three reserve currencies. This is the way that all decentralized, fractional currencies which enable ID registration accept registration fees, equally benefiting all fractional currency holders/liquidity providers when IDs are registered, while providing Verus miners and stakers the identity import fee of 0.02 VRSC.

`*.veth@` IDs have the ability, along with root vrsctest IDs, to make 1:1 ERC20 mapped currencies on Ethereum. They can also be used to create Tokenized ID control tokens (see notes for v0.9.4-1), as all types of IDs can, but if they are used to do that, they will have used their currency creation ability for that rather than an ERC20 mapped currency.

The rest of these release notes are the same as the prior release. Drop by #pbaas-development to start building your vision on Verus and help prepare PBaaS for mainnet!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they

can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your busi...

Assets 6

v0.9.4-1

Asherda
v0.9.4-1
7f145d5
v0.9.4-1

Announcing v0.9.4-1 with the world’s first self-sovereign, tokenized ID control technology, RECOMMENDED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release has no material mainnet changes.

For testnet, this is RC3 of the PBaaS protocol and is required to follow the main testnet chain. RC3 has some significant and important API additions, including the ability for any ID at all, even sub-IDs, to generate a single token as its currency. That token then enables whoever holds it to have another level of revoke/recover power over the ID, in addition to the normal revocation and recovery authorities, which also still work. This technology is now available in the pre-existing APIs, updateidentity (for changing revocation / recovery authorities), revokeidentity, and recoveryidentity. Please see help for details.

In addition, we have continued to harden the protocol and fixed issues with cross-chain currency definition import/export and the new ability to burn reserves into a liquidity basket that were reported in the current testnet.

The rest of these release notes are the same as the prior release. Drop by pbaas-development to start building your vision on Verus while helping get PBaaS ready for mainnet!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again

unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains

with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while loc...

Assets 6

1 person reacted

v0.9.4

Asherda
v0.9.4
e2a52aa
v0.9.4

Announcing v0.9.4, HIGHLY RECOMMENDED FOR MAINNET -- MANDATORY FOR TESTNET USE

This release fixes a mainnet issue that may, in rare cases, cause a node to deadlock.
It also adds the following new features to mainnet:

- Auto TLS encrypted connections between nodes on mainnet and testnet, no action needed, but you can require that all connections be encrypted by starting the daemon with "-tlsenforcement".
- A new "-idindex" flag that enables 3 new APIs to lookup identities by their primary addresses or revocation and recovery addresses, `getidentitywithaddress`, `getidentitywithrevocation`, and `getidentitywithrecovery`.
  Generally improved multithreading within the daemon.
- These improvements allow anyone to now write applications to easily graph or display public relationships between identities, transparent addresses, and transactions without compromising the privacy of private addresses, messages, or transactions between identities using name@:private endpoints, z-addresses or communicating via off-chain channels.

For testnet, this is RC2 of the PBaaS protocol and has some significant and important advances and changes. In addition to a great deal of progress hardening for mainnet, v0.9.4 introduces the following new features to the PBaaS testnet:

- All features above for mainnet

- NFT mapped tokens and tokenized ID control - In v0.9.4, as with v0.9.3, you can use root identities to create any kind of currencies, *.veth IDs to create Ethereum mapped currencies, and what is new in v0.9.4 is that you can now use any type of ID or sub-ID to create a new kind of currency that will always have a 1 Satoshi supply and enable anyone holding it to have power over the ID that issued it. This type of currency is called tokenized ID control and can be treated like a super NFT over all copies of the source ID on all chains that were exported to other chains after the currency was created. Once the Ethereum contracts are upgraded with this capability, *.veth or IDs with root names on the Verus blockchain can be used to create Ethereum NFT mapped currencies, which enable a Cryptopunk, Ethrock, or whatever other NFT you'd like to not only be sent to Verus, held on a revocable/recoverable ID, and also permanently upleveled with ID powers, being cryptographically bound as an ID's permanent control token that can be sent to any chain or to Ethereum and back.
  This unique Verus-only technology and VerusIDs enable new models of complex, atomic and fully decentralized transactions. By using exported IDs across any number of networks and an ID control token, ownership of an unlimited array of assets across an unlimited number of networks can for the first time ever, have ownership transferred in one transaction.

- While NFT mapped tokens are supported on the RC2 PBaaS protocol, they have not yet been implemented in the Ethereum contracts. You will not be able to send NFTs from Ethereum to Verus or PBaaS chains until that work is complete, which we do not expect to take a great deal of time.

- Liquidity currencies now support new methods to burn currency with the following results:
    - `burnweight` - available to centralized currencies only and enables burning the primary currency to raise the reserve ratio weights of all reserves.
    - `burn` for reserve currencies - in prior releases, burn was only used for primary currencies and would reduce the supply of a currency relative to its reserves, increasing the on-chain value of the primary currency. In v0.9.4, you can now also "burn" reserve currencies in a liquidity basket currency, which simply donates that amount of reserve into the currency's reserves without taking any new supply in exchange. This is available to decentralized or centralized currencies.
- Bridgekeeper is now available for permissionless bridge notarization and selectable in the GUI on the mining page. For CLI, we will make a Node application available for testing in the next few days.
- RC1 cross-chain witness and notarization protocol enhanced in RC2 as follows:
    - In RC2, any notarization to be confirmed must be first posted publicly to allow for potential to prove invalidity, and then later confirmed. In the prior protocol, a witnessed notarization could be posted to a chain and accepted when first seen. Upon analysis and discussion with @allbits and others, we realized that the attack vector of even stolen notary/witness keys combined with an in-secret 51% combined hash/stake attack could be prevented if we changed the protocol to first require posting the ready to finalize data and allow notaries who recognize that there is a signature on chain for something they don't agree with (likely stolen key), to simply revoke their identity, which will prevent that notarization from ever being confirmed. This same security feature is supported between all PBaaS chains and has been implemented in the Ethereum contracts for witnesses as well. This new protocol also provides for other types of evidence to prevent finalization, such as proof that there exists a more powerful (most power rule) tip than one witnessed. Enabling decentralized, cross-chain most power rules are an advance in cross-chain protocols that over time will strongly protect censorship resistance qualities of any multichain protocol.
    - The above technology required changes to generalize evidence types between chains, enabling cryptographic enforcement that all notarizations are ultimately determined by miners and stakers and finalized by witnesses who must follow most power (stake+work) rules based on the protocol. Witnesses cannot finalize a chain tip unless alternating miners and stakers have posted 3 consecutive, agreeing, progressing cross-chain notarizations, without intervening disagreement, trading off more certain cross-chain transaction latency for on that depends on miners and stakers, meaning maximum decentralization and a high cost to DoS attack cross-chain, approaching impossible over any significant length of time. This approach makes the incentivized merge mining and bridge mining opportunities, as well as miner/staker participation, automatically part of and also critical to the cross-chain protocol. In v0.9.4, all users on the network may mine the Ethereum bridge via the bridgekeeper tool on the mining page in the GUI.
    - With this new protocol, all PBaaS chains on testnet will consider witnessed / finalized tips final and will not unwind past those witness points or follow a chain

that disagrees with a known finalized tip. When this protocol goes live on mainnet, and for any PBaaS chain whose community opts to use the Komodo notary service, Komodo notarizations will continue to ensure a stable chain tip, but no matter what other notarizations are present, PBaaS chains will always follow their proof of power protocol and lock to their witnessed tips as well.

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

The rest of these release notes are the same as the prior release. Drop by pbaas-development to start building your vision on Verus while helping get PBaaS ready for mainnet!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Curren...

Contributors

-

allbits
Assets 6

v0.9.3

Asherda
v0.9.3
9e8227b
v0.9.3

Announcing v0.9.3, NON-MANDATORY FOR MAINNET -- MANDATORY FOR TESTNET USE

This release fixes one issue on mainnet where a node using -connect=”networkaddress” may still attempt to connect to other nodes.

This update includes the first protocol release candidate for mainnet PBaaS, VerusID protocol upgrades, and Verus DeFi. In addition to some functional fixes and full web site support for mapped currencies and simple commands for mapping Verus currencies to any ERC20 currencies on Ethereum with provable, auditable 1:1 mapping and cross chain send/receive. In addition this new protocol upgrade changes some parameters of the Verus Proof of Power consensus algorithm, enabling smoother starts, restarts, and general operation of PBaaS chains, regardless of their native

currency supply or emission schedule. As these consensus parameter changes would require either a testnet fork, which would not be relevant for the mainnet upgrade or a reset, accessing the v0.9.3 testnet requires that you reset your local testnet by deleting the vrsctest and pbaas data folders from the locations described below.

As part of this release, we will be launching the new Ethereum bridge and enable full use of Ethereum mapped currencies on the Goerli Ethereum testnet (note that we were previously using Rinkeby). This makes it easy for any currency definition to become a fully decentralized, auditable stable coin, simply by mapping to DAI, USDT, USDC, or another stable coin on Ethereum. In addition, the new website for Metamask support and use from Ethereum now displays all mapped currency options, even those added when someone exports a currency definition to Ethereum, either as a mapped currency or for using any Verus or PBaaS currency as an ERC20.

This release also resolves an issue with setting max preconversion amounts on currency launches and addresses a number of minor protocol issues found as a result of the active community testing support we are seeing on testnet.

v0.9.3 RESETS TESTNET AND ADDRESSES ALL KNOWN ISSUES FOUND DURING TESTING ON THE NETWORK. FROM HERE, WE HOPE ALL COMMUNITY MEMBERS WILL JOIN IN TRYING OUT ALL FEATURES OF TESTNET TO HELP ENSURE THAT THE FINAL MAINNET RELEASE, WHICH WE ARE WORKING TOWARDS AS SOON AS IT IS CONSIDERED READY AND HARDENED, WILL BE BOTH SMOOTH AND WORLD-CHANGING.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Platform Support

- Debian 9 (Stretch) is no longer supported

PROTOCOL RELEASE CANDIDATE RC1

With v0.9.3, The Verus PBaaS protocol, including networking, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi, is considered release candidate RC1 for the coming mainnet upgrade. The core developer focus now remains on protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.3, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.

More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at 100 USD worth of ETH to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.3 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute

transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next ...

Assets 6

v0.9.2-3

Asherda
v0.9.2-3
afc6987
v0.9.2-3

Announcing v0.9.2-3, NON-MANDATORY, HIGHLY RECOMMENDED FOR MAINNET -- CRITICAL FOR TESTNET USE

This update enables full use of Ethereum mapped currencies (Rinkeby when the new bridge launches), which provides the ability for any currency to become a fully decentralized, auditable stable coin, simply by mapping to DAI, USDT, USDC, or another on Ethereum. In addition, the new website for Metamask support and use from Ethereum now displays all mapped currency options, even those added when someone exports a currency definition to Ethereum, either as a mapped currency or for using any Verus or PBaaS currency as an ERC20.

This release also resolves two PBaaS chain launch issues, one that prevented no-reward chains with initial emission from being accepted and one that could allow a chain launch to go forward, even when a co-launched liquidity basket failed to get enough participation. Thanks to increased participation on testnet use, we are getting continued excellent coverage and resolving issues, which are increasingly edge conditions as we go. discovered on both mainnet and testnet and on testnet fixes an error creating transactions for sub identity registrations on fractional reserve currencies. If you already went through the testnet reset process on any version since v0.9.2, you do not need to reset testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it

will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds
```

```
verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value a...

Assets 6

v0.9.2-2

Asherda
v0.9.2-2
`79d6171`
v0.9.2-2

Announcing v0.9.2-2, NON-MANDATORY, HIGHLY RECOMMENDED FOR MAINNET -- CRITICAL FOR TESTNET USE

This update improves sync speed in some edge cases on both mainnet and testnet and on testnet fixes an error creating transactions for sub identity registrations on fractional reserve currencies. If you already went through the testnet reset process on any version since v0.9.2, you do not need to reset testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a

head start on using Verus's transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {"proofprotocol":2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`

- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which ...

Assets 6

v0.9.2-1

Asherda
v0.9.2-1
34c2fb5
v0.9.2-1

Announcing v0.9.2-1, NO FUNCTIONAL CHANGES TO MAINNET FROM v0.9.2 -- CRITICAL FOR TESTNET USE

This update fixes an error in creating transactions for sub identity registrations on fractional reserve currencies and it fixes a synchronization issue that could prevent some clients from fully syncing testnet. The rest of these release notes are the same as the prior version.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY

IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.

More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a

parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary...

Assets 6

v0.9.2

Asherda
v0.9.2
`5c91a86`
v0.9.2

Announcing v0.9.2, OPTIONAL, HIGHLY RECOMMENDED FOR MAINNET, MANDATORY FOR TESTNET USE - THIS IS A BIG ANNOUNCE, BUT IF YOU CARE ABOUT THE PBAAS UPGRADE, WE RECOMMEND READING IT ALL.

v0.9.2 RESETS TESTNET AND ADDS A COMPLETE SET OF NEW CAPABILITIES PREVIOUSLY CONSIDERED STRETCH GOALS THAT ENABLE BRANDED SUB-ID ISSUANCE WITH BOTH DECENTRALIZED AND CENTRALIZED OPTIONS AS WELL AS THE FIRST DISRUPTIVE TECHNOLOGY TO SOLVE FOR BRANDED LOYALTY IDENTITIES WITH NOVEL SOLUTIONS FOR ROYALTY AND IP LICENSING BUSINESS MODELS.

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

FOR MAINNET (AND TESTNET), v0.9.2 PROVIDES ENHANCED PRIVACY FEATURES FOR IDENTITY CREATION

PROTOCOL FEATURE COMPLETE

With v0.9.2, The Verus network, public blockchains as a service (PBaaS), VerusID extensions, and Verus DeFi protocols are considered feature complete for the coming mainnet PBaaS upgrade. The

core developer focus now is protocol hardening and enabling usage first, followed by user experience upgrades and supporting infrastructure such as identity and login support for lite mode, mobile and web extension wallets as well as merge mining for pool servers.

NEW 3D CROSS-CHAIN LIQUIDITY NETWORK VISUALIZER (GUI FOR TESTNET)

In v0.9.2 Verus Desktop, there is a new 3-dimensional cross-chain network visualizer, which shows all currency conversion and fractional network connections both on chain and cross-chain, with the ability to move around and zoom in and out in 3 dimensions. To access the visualizer, click on “Visualize Network” on the Metaverse tab of Verus Desktop. If you’re interested in seeing the visualizer now, here’s a link directly to the visualizer as part of a recent video members of the community helped make (https://youtu.be/CkgUVjx_g1M?t=43).

NEW PBAAS CAPABILITIES

All PBaaS capabilities, including for application development, are available now on the new Verus testnet. Our goal for interoperability on this new testnet is to launch the Ethereum/Rinkeby bridge in the next couple days. Now would be a good time for development capable projects hoping to get a head start on using Verus’s transformative technologies to do a deep dive on the Verus testnet. These new capabilities in v0.9.2, as well as all those we are familiar with, available nowhere else, will be on mainnet SOON(™):

All fractional and centralized {“proofprotocol”:2} currencies can now issue as a decentralized operation, or with centralized currencies, sell, IDs using their own currency as the payment currency, with options for price, referrals, permission from the currency identity required, which means the transaction must be created or co-signed by the currency identity, or even referrals required, which means that an identity registration must have a referral and co-signature of a referring ID that is either the currency ID or another ID from that currency namespace.
For fully decentralized fractional currencies, ID registration is paid for by burning the specific fractional currency, reducing the supply, and raising the on-chain value of the primary currency relative to its reserves by reducing the current on-chain supply for everyone.
The minimum price of ID registration for fractional currencies can be set in either the fractional currency itself, or in one of its reserves, for example USDC or DAI.
More than the minimum price can be paid for ID registration, which may also include VDXF key and contract bindings to allow for multiple types of content or IP licensing in a single ID namespace.
Each currency, either centralized and/or fractional, can have multiple levels of referral support (limit of 5 levels), meaning that the ID registration fee will be reduced when referred and part of it will be divided evenly among past referrers. The remaining amount of the ID registration fee will be burned for decentralized currencies, or be sent to the currency ID for centralized currencies.

While Native IDs, those registered directly from a blockchain as the name parent, will be able to launch currencies on that blockchain, and if on Verus, actual independent blockchains, IDs issued by fractional or centralized currencies cannot launch currencies, but will still be valuable as revocation and recovery IDs for all other VerusIDs as well as for personalized on-chain and cross-chain addresses with all identity and NFT features, for proofs, for brands, contract binding, on-chain

marketplace trading, or login, and will have the ability to cross-prove common ownership with other IDs as well.

The exception to non-native IDs creating currencies, is that *.veth gateway IDs will be available for decentralized registration from the Ethereum bridge converter and will be able to create Ethereum mapped currencies. Registration of .veth IDs will be priced at a level to prevent misuse and support bridge liquidity, with all fees besides the 1 VRSC import fee, which goes into the mining and staking fee pool going to the bridge converter liquidity basket.

All of these features are currently enabled on the v0.9.2 new testnet. Stay connected on the #pbaas-development channel and participate in the launch at no actual cost as a testnet liquidity provider and get some nice testnet IDs, launch your own currencies, or start your exclusive ID-restricted club, governance group, or business.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
```

```
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to who...

v0.9.1-2

Asherda
v0.9.1-2
6c5c68b
v0.9.1-2

ANNOUNCING PBAAS TESTNET UPGRADE v0.9.1-2, OPTIONAL FOR MAINNET USERS -- MANDATORY FOR TESTNET USERS

v0.9.1-2 RESETS TESTNET -- THERE ARE NO MAINNET CHANGES

IF YOU HAVE ALREADY USED TESTNET WITH VERSIONS PRIOR TO v0.9.1-2, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED BELOW.

Testnet Featuring Verus PBaaS

The most powerful and interoperable currency, blockchain, and ERC20 launch, scale, and MEV-resistant DeFi platform to exist, all with no programming required!

Verus PBaaS Features Live on v0.9.1-2 Testnet

- Provable or pseudonymous, identity-based currency, liquidity pool, NFT and multi-blockchain, fully decentralized network.
- On-chain 100% decentralized, launches with fair launch and/or crowdfunding options, on-chain auto-refund option for missed targets, automatic Ethereum ERC20 contract deployment, auto-created, MEV-resistant, liquidity baskets and even auto-bridged, merge mineable, fully independent rent-free PBaaS blockchain launches.
- All DeFi AMM conversions are verified via mining and staking as part of the L1 consensus protocol. All conversions in a single liquidity basket are calculated simultaneously for all transactions in any given block, meaning all participants get the same price in all directions of conversion, with a minimum conversion fee of 0.025% and a maximum of 0.05%.
- All currencies and identities are primitives at L1 and are validated and verified on UTXO transactions, which check all inputs and outputs just as single currency L1, such as Ethereum or Bitcoin check the native currency inputs and outputs of transactions.
- All currencies on all independent connected chains, once launched can be sent back and forth to other multi-currency capable networks (currently other independent PBaaS chains and Ethereum as ERC20s).

- Currencies on Verus can also be defined as "mapped currencies", which map 1:1 to an existing currency on Ethereum (eg. DAI), and can then be sent from Ethereum with simple transactions and received and used as the new "mapped currency", or sent back to Ethereum and used there as the original currency.
- Currency launches can raise funds in 3 ways. Each of these fundraising options creates a currency that is not 100% backed, and has a price that responds to market forces. 100% backed currencies do not change their relative price to the underlying basket reserve when they are converted to or from reserves. Fractionally backed currencies, which result from fundraising or endowing launch grants/pre-allocations to DAOs or entities such as foundations, whether time-locked or not, take their funding from a percentage of the reserve backing. Except in the case of a new blockchain launch, which creates new native currency for operating the proof of stake of the blockchain, any currency that uses fundraising options will be a fractionally backed currency which will appreciate when more people convert to it and depreciate when more convert from it. If a launch is refunded due to minimum participation options that are not met, no fundraising is received:
    - Pre-launch discount - all participants contributing to the launch before it goes live on its start block get a discount to the initial on-chain price of the currency. This is usually accompanied by a maximum cap on pre-conversion participation to ensure that the currency would still be in demand when it goes live.
- Pre-launch carve-out - this enables a percentage of each of the initial participation currencies to be carved out of the launch and total reserve ratio in a price neutral manner and sent directly to the launching identity upon successful launch.
- Pre-allocation - this enables some of the newly launched currency to be pre-allocated to one or more identities. If the currency is fractional, this happens in a price neutral manner, and again is subtracted from the reserve ratio. Recipients can optionally be time locked IDs for vesting or unlock periods.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.Komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which t...

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs

- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-
-
-
-
-
-
-
-
-
-
-
  -
  -
  -
- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

-
-
-
- Code

- Issues 21
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Releases: VerusCoin/VerusCoin

Releases Tags

v0.9.9-1

Asherda
v0.9.9-1
`08a713f`
v0.9.9-1

Announcing v0.9.9-1, MANDATORY FOR TESTNET, no mainnet changes

This release implements a fix for communication with BridgeKeeper for the Testnet Goerli Ethereum Bridge, as well as edge-case notarization improvements. We need every testnet participant to upgrade to this version as soon as possible to resume notarizations for the ETH bridge and the pbaas chains. Once enough people upgrade, the testnet upgrade can be broadcasted via oracle and we can get closer to getting pbaas on mainnet.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents,

marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Veru...

Assets 6

v0.9.9

Asherda
v0.9.9
`ad8fb8d`
v0.9.9

Announcing v0.9.9 - PBaaS Mainnet Preparation Release, MANDATORY FOR MAINNET AND CONTINUED TESTNET USE

What's New

There is so much new and important in v0.9.9, and we are so close to the mainnet PBaaS release that this is a hard announcement to keep to a digestible size. Key points:

1. **Preparing for #LaunchPBaaS**: v0.9.9 is MANDATORY, meaning that you will need to upgrade to this release ASAP to stay reliably connected to mainnet after the first preparatory PBaaS upgrade is triggered, both for testnet and mainnet. While the current deadline for being upgraded is in 1 week, it is a soft fork that we would like to activate as soon as possible, and if we can believe that we have more than 50% of the validating network updated earlier than 1 week, we will trigger the activation with our Oracle Notification Technology described in 2. **(Mainnet + Testnet)**

2. **Enabling Seamless Decentralized Network Upgrades**: This release includes a Verus invention of consensus-capable oracle notifications built into the core protocol, which enable dynamic coordinated consensus changes that can be triggered at a time decided after the actual release which enables them, all without any compromise on decentralization. The Verus Community may decide that we are ready to go earlier than one week on this upgrade and activate it, even before its scheduled time. The good news is that this first test upgrade will primarily affect testnet and prepare for the mainnet PBaaS upgrade, so it is a perfect test case for a mainnet test run before we #LaunchPBaaS. This release actually has the ability to upgrade mainnet to PBaaS, but we believe it is prudent to have at least some time on this new protocol on testnet before the v1.0 release, which will upgrade mainnet. If we make no additional changes to the protocol in this release, which we do not intend to do at this time, this version may be able to follow the v1.0 upgrade. If we determine that it cannot, the oracle upgrade technology will enable the daemon to recognize that it is not capable and request that the user upgrade to a newer version without requiring a bootstrap for the upgrade. **(Mainnet + Testnet)**

3. **This upgrade fixes all known testnet issues with no reset**: Using v0.9.9, you should be able to sync fully and resume all cross-chain operations on the Verus test network, if you were not able to perform cross-chain operations or sync to the network on the prior version. As mentioned, someone seems to have mined with an intermediate version along the way on testnet and made the network unavailable without a bootstrap on the older daemon. This version should fix that issue as well as any cross-chain difficulties. Auto-notarization will be enabled on testnet in sync when the testfork executes. Until then, all witnessed cross-chain operations should be able to resume, if they have had difficulty. **(Testnet until the #LaunchPBaaS release)**

4. **Bridging Blockchain Networks Without Witnesses**: v0.9.9 is the first release of any blockchain protocol anywhere that we know of which enables cryptographically provable, cross-chain bridging across Verus and all PBaaS and other connected blockchains in the network, either combined with or independent of the need for notary witnesses. With notary witnesses, cross-chain transactions complete faster, but they are also checked against cryptographic evidence, making it difficult to impossible for anyone, including notary witnesses to take any action, even colluding with validators that would go against the wishes

of the most chain power. We have been referring to this new cross-chain consensus technology as “auto-notarization”, as that is what it actually is, but it would also be accurate to call it Proof of Proof of Power, as that is how it works. For purposes of ensuring everyone understands the state of the technology, it is fully functional and we have taken great care to ensure it is sufficiently secure, but we have not yet had the protocol proven to an academic standard. Because of that, while it provides a security fall-back against misbehaving notary witnesses and will be fully functional and we believe sufficiently secure without notary witnesses on release, we currently recommend toconsider the protocol unproven at this time and that any serious PBaaS chains be launched with operational notary witnesses until further notice.

Using `“notarizationprotocol”:1`, which is also the default, a chain launch can specify notary witnesses that will be used when they do their job. Even if they stop witnessing for any reason, the protocol will fail-over to auto-notarization, ensuring that cross-chain transactions still function as expected, even if just for people to move their assets to another blockchain, although significantly more slowly and based solely on cryptographic proof and challenge resolution between merge miners and stakers. This cross-chain proof and challenge protocol operates even when witnesses are also operating and if miners and stakers continue to prove a more powerful chain than the one witnesses represent, the most powerful chain can still be confirmed. **(Testnet until the #LaunchPBaaS release)**

5. **Expanding L1 VerusID APIs**: This release provides a major enhancement to the identity content multimap support with two new APIs, `getidentityhistory` and `getidentitycontent` that provide real-time mempool access and also introduce self-sovereign delete operations to identity content that can be used by applications, making it easy for applications to index, organize, and aggregate an unlimited amount of user content referenced by an identity and made available to the application. This is the technology we used for oracle upgrade notifications, as all of this is available at L1, making it possible to imagine future protocols that can deal smoothly with diverse community opinions worldwide, but always respect the sovereignty and decisions of all members on the network. **(Testnet until the #LaunchPBaaS release)**
6. **Introducing Restricted VerusID Staking for PBaaS Chains**: This release enables the stretch goal of restricted ID staking as an option for launching PBaaS chains. This option means that on such a PBaaS chain, only funds controlled by IDs with a parent of that PBaaS chain may stake blocks on that chain. When combined with restricted forms of ID issuance, whether approval or referral required and/or referral rewards, this capability opens up an entire new area for public/private blockchains and applications, as well as government, voting security, corporate, or organizational use cases.**(Testnet until the #LaunchPBaaS release)**

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even con...

Assets 6

1 person reacted

v0.9.6-2

Asherda
v0.9.6-2
86d0881
v0.9.6-2

v0.9.6-2, CRITICAL UPDATE FOR MAINNET
This release contains a critical update for the Verus daemon. This version does not support testnet, but a future version will be released shortly to support it. For more information on running testnet, please refer to the previous release or the next one.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an

unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and

a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will

automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the Verus blockchain network is fractal, enabling an unlimited number of simultaneously operating, interoperable blockchains....

Assets 6

v0.9.6-1

Asherda
v0.9.6-1
1b02282
v0.9.6-1

v0.9.6-1, RECOMMENDED UPDATE FOR MAINNET, CRITICAL FOR CONTINUED TESTNET AND ETHEREUM BRIDGE (GOERLI) USE

What's New
THERE IS NO TESTNET RESET, AND ALL OTHER NOTES REMAIN THE SAME AS LAST RELEASE.

v0.9.6-1 has no mainnet changes and fixes the following testnet issues:

- Fault when initiating mining and staking on a small number of machines
- Sync from scratch issues on testnet

This release fixes all reported issues in all community testing. There are no protocol changes, and no known changes required. All network functions should work as intended.

Verus PBaaS and DeFi RC6 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet

release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don't expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (3 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

If you have reported a bug, since the bounties for the 0.9.5+ versions were announced, you are on the list, and someone from the Foundation will reach out to you in the near future.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release this coming week, and start building your vision today!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along w...

Assets 6

2 people reacted

v0.9.6

Asherda
v0.9.6
`7f971ed`
v0.9.6

Announcing v0.9.6, HIGHLY RECOMMENDED UPDATE FOR MAINNET, MANDATORY FOR NEW, RESET TESTNET AND ETHEREUM BRIDGE (GOERLI) USE

v0.9.6 RESETS TESTNET AND COMPLETES ALL PROTOCOL UPDATES FEATURES AND CAPABILITIES EXPECTED FOR THE MAINNET PBAAS UPGRADE. WE DO NOT HAVE ANY NEAR TERM PLAN FOR ANOTHER TESTNET RESET AFTER THE v0.9.6 UPGRADE.

IF YOU HAVE USED TESTNET WITH VERSIONS PRIOR TO v0.9.6, MAKE SURE TO DELETE AND CLEAR ANY EXISTING TESTNET DATA FOLDERS AS DESCRIBED IN THE RELEASE NOTES LINKED BELOW.

What’s New

For mainnet, v0.9.6 fixes the following issues:

- An error in fee calculation that can result in nodes having difficulty accepting large, complex transactions with many z-outputs into their mempools.
- Marketplace offers were not showing up properly under getoffers.
- When using the `-privatechange` wallet option with a private change address, attempts to make marketplace offers could fail.

For testnet, v0.9.6 adds the following:

- Caching and performance improvements for advanced PBaaS functions, providing enough performance to keep high transaction bandwidth and/or cross-chain throughput under load for each chain.

- Most hardening required for the mainnet release.
- **The Verus/Ethereum Bridge!** v0.9.6 includes the upgraded Ethereum bridge to the Goerli testnet live with all bridgekeeping, Ethereum Gas price tracking, and reward incentive capabilities for miners and stakers. Rewards for bridgekeeping are delivered both on Verus and on the Ethereum blockchain. Access the Ethereum bridge website (https://ethtest.verus.services) to move over some Goerli Ethereum or USDC, and participate in the Bridge.vETH liquidity basket launch to hold and earn fees on a basket with vETH, USDC, and VRSCTEST used as the fee converter for all cross chain transactions between Ethereum's Goerli and VRSC testnets.

v0.9.6 also includes support for new parameters `-arbitragecurrencies=["VRSCTEST","vETH"]` and `-arbitrageaddress=valid R-address, pubkey, or ID`. They're exciting, but don't try to use them just yet. They are for testing right now, and are preparing for an "auto-arbitrage" feature already supported in the protocol, but which we hope to have in the daemon before mainnet PBaaS activation that combines decentralized on-chain marketplace offers for currency with liquidity basket import processing to provide miners and stakers the opportunity to commit some amount of funds from their node wallet in common currencies to spot and leverage guaranteed arbitrage opportunities when making a block. These opportunities arise between orders on chain and liquidity basket pricing that if leveraged to earn, actually result in P2P crowdsourcing of liquidity for all baskets across the network, both from the blockchain itself as well as miners and stakers, while benefiting all participants in the following ways:

- Miners and stakers can capture guaranteed arbitrage opportunities and arbitrage with any currency that they choose from their wallets.
- Blockchain users can set limit orders and have them filled if they are near the market price for a currency, or if the market moves in their direction.
- LPs and users of liquidity baskets for conversion get as close to market prices as possible on their conversions, never releasing funds from a basket before an on-chain, MEV-resistant auction takes place for the pricing of those assets.

Although this release is likely to be robust enough for a mainnet, we have come this far with quality and will not rush for a few days earlier upgrade. We still plan to review and complete a small number of hardening items before PBaaS 1.0 is ready for its mainnet release.

That said, this release fixes all reported issues in all community testing. Community developers, including myself are grateful for all of the testing many have contributed to to date. It has made it possible to get incredible test coverage across all network capabilities, and we appreciate all of your participation and pushing of boundaries. This release is expected to be the final PBaaS protocol that will be released on mainnet. There are no known changes required, and all functions should work as intended.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Verus PBaaS and DeFi RC6 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don't expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that ...

Assets 6

v0.9.5-2

Asherda
v0.9.5-2
bc894ba
v0.9.5-2

Announcing v0.9.5-2, RECOMMENDED UPDATE FOR MAINNET - CRITICAL FOR CONTINUED TESTNET USE

YOU MUST BE UPDATED TO v0.9.5 OR LATER BY BLOCK 2291830, AT APPROXIMATELY 11:00 PM UTC, SATURDAY, NOVEMBER 19th, 2022 TO RELIABLY STAY SYNCHRONIZED TO THE VERUS MAINNET. WHILE THERE MAY BE A MAINNET PBAAS UPGRADE BEFORE THAT DATE, WE RECOMMEND THAT IF YOU HAVE NOT UPGRADED YET, THAT YOU DO SO NOW.

As with all v0.9.5-* releases, this release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022.

What’s New

For mainnet, this release labels spendable transactions for the getaddressutxos RPC API, which is used by Verus Mobile. This update is critical for users that have not upgraded to v0.9.5+ before block 2291830.

For testnet, v0.9.5-2 introduces minor notarization fixes that will be important for those intending to launch and witness PBaaS chains.

Ethereum Bridge

Although its deployment has been delayed, there are no technical blockers, and we still expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services/, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the pbaas-development channel, help the community prepare for the imminent mainnet release of the PBaaS upgrade and start building your vision today!

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don’t expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the

blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh

private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional bas...

Assets 6

v0.9.5-1

Asherda
v0.9.5-1
9dc792b
v0.9.5-1

Announcing v0.9.5-1, CRITICAL UPDATE FOR MAINNET MANDATORY FOR CONTINUED TESTNET USE

As with the last release, this release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022. YOU MUST BE UPDATED TO v0.9.5 OR LATER BY THEN TO RELIABLY STAY SYNCHRONIZED TO THE VERUS MAINNET. WHILE THERE MAY BE A MAINNET PBAAS UPGRADE BEFORE THAT DATE, WE RECOMMEND THAT IF YOU HAVE NOT UPGRADED YET, THAT YOU DO SO NOW.

Please make sure that if you do choose to wait for the possibility of a mainnet PBaaS upgrade before this activation that you do upgrade to the latest release before Saturday, November 19th.

What’s New

For mainnet, this release has no material changes over v0.9.5, but is still considered critical, as mainnet users should be upgraded to v0.9.5+ before block 2291830.

For testnet, v0.9.5-1 fixes an issue discovered on testnet where duplicate ID exports blocked cross-chain processing temporarily and also introduces a protocol improvement designed to make the Verus anti-MEV DeFi protocol even more resistant than what we believe was already the leading algorithm to multi-block attempts at MEV. The protocol change makes the following adjustments:

Conversions and cross-chain transfers still roll-up into exports that are collected from one or more blocks of transfer transactions, but to determine when a block is the last of the prior export or first in the next requires a random bit pulled from later blocks.
The last miner or staker to complete a block of an export retroactively receives 10% of all fees from processing that export, including all conversions, once the rollup is processed as an import. This reward used to go to the miner or staker who processes the export, which is no longer the case.
Import transactions can now include up to n/2 arbitrage transactions, where n is the number of reserves in a liquidity basket currency.

These changes together serve to disincentivize any miner or staker from attempting to exclude a large set of transactions from a block and front run all of them, as they will have an even chance of having all of those excluded transactions put back in with their transaction, turning potential gain into potential loss. They would be better off saving their working capital to arbitrage imports by joining transactions, bringing them close to market, and earning money without risk of loss. In addition, if they do end up putting more value of transactions in a block that caps the export rollup, they will reliably earn meaningful rewards without risk of loss in that way as well.

Ethereum Bridge

Although its deployment has been delayed, we expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don’t expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people (7 still unclaimed) who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the

blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables id...

v0.9.5

Asherda
v0.9.5
`91fd525`
v0.9.5

Announcing v0.9.5, CRITICAL UPDATE FOR MAINNET AND TESTNET

This release updates the Komodo notaries to season 6, with an activation height of 2291830, at approximately 11:00 PM UTC, Saturday, November 19th, 2022. You must be updated to v0.9.5 or later by that time to reliably stay synchronized to the Verus mainnet. We are also working to get a mandatory update release during these next couple weeks, which will have the Verus PBaaS mainnet upgrade finalized as well. If you would prefer to wait for the PBaaS mainnet upgrade and upgrade once, we cannot guarantee that the PBaaS mainnet release will be available in the next couple weeks, but we are working to make that a reasonable choice as well. Please make sure that if you do choose to wait to upgrade that you do upgrade to the latest release before Saturday, November 19th.

New Signature Capabilities on Mainnet & Testnet

In addition to continued support for all of the `signmessage`, `signdata`, and `signfile` APIs, as well as their verify* counterparts, v0.9.5 introduces 2 new APIs and CLI commands, `signdata` and `verifysignature`, that encompass all of the capabilities of previous signing and verification APIs and add new capabilities for advanced signatures and verification. These new advanced capabilities were designed for community work by @allbits, designing a more powerful and flexible framework for NFTs and ID-bound legal rights and agreements. In addition to digital signing of documents or information using an ID, the new APIs enable easily binding together multiple sources of content, documents, identities, and VDXF keys together into one verifiable signature and then being able to verify that signature, including all of its specific bindings.

Verus PBaaS and DeFi RC5 - BUG BOUNTIES

At this time, we are not aware of any bugs or errors in the DeFi, ID, or PBaaS cross chain protocols on this version of testnet. We do not anticipate any changes to the PBaaS protocol before mainnet release, and we hope you will join us on testnet trying out all of the incredible capabilities and looking for any potential, remaining issues that can possibly be found before mainnet release. While we don’t expect it to be easy to find bugs in the protocol, The Verus Coin Foundation is offering a bounty of at least 500 VRSC (possibly more for security related reports) for the first 10 people who are first to report any actual protocol bug before mainnet release, meaning failure of a properly executed command or API on a functioning VRSCTEST or PBaaS chain to function as intended. If anyone is first to find and describe the exploit for an exploitable security hole in any part of the protocol, the bounty will be a minimum of 1,000 VRSC, and for a serious potential security issue, 10,000 $VRSC.

Emoji IDs and 🎃Chain

In the spirit of this release day, we have taken advantage of VerusIDs and currency emoji support and created a new, merge mineable PBaaS chain, 🎃. It has been said that those who merge mine 🎃chain at midnight will disappear into the crypto-metaverse, never to be seen again, become VerusID avatars, and be forever doomed to being bought and sold on the VRSCTEST marketplace for all eternity (or until the next testnet reset). YOU HAVE BEEN WARNED!

We expect the Ethereum bridge to be brought back up on the new Verus testnet within the next day or two. When it is up, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services, and use your metamask to convert or send cross-chain to the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet release in the next couple weeks and start building your vision today!

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a

parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-sig...
```

Contributors

-

allbits
Assets 6

v0.9.4-5

Asherda
v0.9.4-5
25ed684

v0.9.4-5

Announcing v0.9.4-5, RECOMMENDED FOR MAINNET AND TESTNET

This release fixes an RPC API issue on mainnet and testnet that can cause a daemon fault when the getaddressmempool API is used. This fault while running is only caused by non-typical API, insight explorer, or CLI use, but if it does happen, may result in the need to rescan or resync your wallet. We know of no one on mainnet affected by this issue. We recommend that all users upgrade.

This release also includes a new capability for testnet, which was pioneered by CHIPS developer @biz, who worked to modify and test the Verus PoP algorithm to support 10 second block times, which the CHIPS chain uses to enable its game application focus. These new parameters will enable the CHIPS blockchain to move to being an interoperable, merge mineable PBaaS chain with a 10 second blocktime when PBaaS goes live on mainnet. To learn how to test the new CHIPS test chain on testnet, visit the #pbaas-development channel, as the support in v0.9.4-5 is not as complete as it will be in the next and hopefully final testnet reset. In that next version, blocktime and the pow averaging window for any PBaaS chain's PoP consensus will be able to be configured between 10 seconds and 2 minutes in addition to its notarization frequency as part of the chain definition, enabling a greater variety of PBaaS use cases, transaction throughput, and application capabilities.

As a reminder, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services , using your metamask, converting or sending cross chain through the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. We encourage you to drop by the #pbaas-development channel, help the community prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`

- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute

transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an inter...

Contributors

-

biz
Assets 6

v0.9.4-4

Asherda
v0.9.4-4
538c91a
v0.9.4-4

Announcing v0.9.4-4, UNCHANGED FOR MAINNET -- RECOMMENDED FOR CONTINUED TESTNET USE, MANDATORY FOR CONTINUED TESTNET MERGE OR BRIDGE MINING

This release has no material mainnet changes.

This release primarily upgrades the ETH bridge and also fixes a PBaaS cross-chain notarization issue that made miners and stakers consider themselves ineligible to enter cross-chain notarizations. There are no actual protocol changes.

The ETH bridge upgrade introduces the ability to enter notarizations between Ethereum Goerli and VRSCTEST using bridgekeeper without an Ethereum private key in your conf file. This will enable you to still mine and potentially earn by agreeing or disagreeing with notarizations mined into the Verus blockchain, without needing to make transactions on or spend Ethereum for fees. This should also work with a free Infura account.

As a reminder, you can access the Verus <-> Ethereum testnet bridge here: https://ethtest.verus.services , using your metamask, converting or sending cross chain through the Verus network from Ethereum's Goerli. To use these features you will need to get some Goerli Ethereum, either from a faucet or via the Verus testnet liquidity baskets on Verus.

The rest of these release notes are the same as the prior release. Drop by the #pbaas-development channel to help us prepare for mainnet and start building your vision today!

Tokenizing ID control (next generation NFT):

The currency definition have flags OPTION_NFT_TOKEN + OPTION_TOKEN, and a max supply of 1 satoshi that is either pre allocated or pre-converted to. If the token is pre-allocated, then the maximum pre-conversion must be 0.

```
verus -chain=vrsctest definecurrency
'{"name":"ID","options":2080,"preallocations":
[{"ControlTokenRecipient@":0.00000001}],"maxpreconversion":[0]}'
```

Testnet Reset

To reset your testnet make sure Verus is closed (and no testnet daemon running) and delete the following directories, then restart the testnet daemon (or relaunch Verus Desktop, deactivate verustest and re-add verustest native):

- Linux: `~/.komodo/vrsctest`, `~/.verustest`
- Mac OS: `~/Library/Application Support/Komodo/vrsctest`, `~/Library/Application\ Support/VerusTest`
- Windows 10: `%AppData%\Roaming\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest` or `%AppData%\Komodo\vrsctest\`, `%AppData%\Roaming\VerusTest`

Creating an identity with a fractional currency as its parent

`registernamecommitment` now takes two more positional arguments to specify a currency parent and a funding address. Use quotes `""` to leave fields blank, the example below specifies a parent currency, vrsc-btc , but no referrer. We're now able to use z_addresses to fund the name commitment and identity registration

```
# verus -chain=vrsctest registernamecommitment name controladdress referral
parent sourceoffunds
```

```
verus -chain=vrsctest registernamecommitment subID
RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr "" vrsc-btc
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

Specify the parent in the identity definition. Enter `false` for `returntx` to sign and submit the id registration, `0` for the `feeoffer` to use the default fee, and the funding identity, transparent address, or z-address

```
# verus -chain=vrsctest registeridentity '{ID registration with name
commitment}' returntx feeoffer sourceoffunds

verus -chain=vrsctest registeridentity '{"txid":
"67635331cbccb7a2cbf408a9e97b3f8986133964e0315a8b9fd237a5fd95ac8f","namereservat
ion": { "version": 1,  "name": "ID",  "parent":
"i84mndBk2Znydpgm9T9pTjVvBnHkhErzLt", "salt":
"b7070f2ca7495e49c85ab41b5a368150e2c217be6d08cc4102a1b682cddb6f01", "referral":
""},"identity":{"primaryaddresses":
["RDnf7mH7RQki9b7PqdBD2Er6WXv3DTawGr"],"minimumsignatures":1,"name":"ID","parent
":"vrsc-btc@"}}' false 0
zs1s2mteau9tcalvk55cnepw3aq7dr6w7f447pqqkxczat3a02208d3ersx60wz9srw3nkd25ppfny
```

If a currency's ID issuance require permission from the currency's identity then it must sign the name commitment and identity registration. Either use the parent identity to fund those transactions, or receive a raw transaction to give the identity owner to sign by setting `returntx` to `true`

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

Exporting an ID to a PBaaS chain

```
verus -chain=VRSCTEST sendcurrency "*"
'[{"address":"IDNAME@","exportto":"PBaaSChainName","exportid":"true","amount":10
0,"currency":"vrsctest"}]'
```

Signing transactions from multi-signature IDs (testnet and mainnet)

Create transaction, get raw transaction data:

```
verus sendcurrency <multi-signature-ID>@
'[{"address":"<destination_address>","amount":<transaction_amount>}]'
verus z_getoperationstatus <operation_id_returned_by_sendcurrency>
```

Take the raw hex transaction data provided by z_getoperationstatus to each additional wallet(s) containing the additional signing addresses/IDs:

```
verus signrawtransaction <raw_hex_transaction>
```

After the last necessary signature is applied, broadcast on the network using:

```
verus sendrawtransaction <raw_hex_signed_transaction>
```

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking...

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Skip to content

-

- Pricing

Sign in
Sign up

VerusCoin / **VerusCoin** Public
forked from miketout/VerusCoin

- Code
- Issues 21
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Releases: VerusCoin/VerusCoin

Releases Tags

v1.0.5

Asherda
v1.0.5
8022fec
v1.0.5 Latest

Announcing v1.0.5 - MANDATORY MAINNET UPGRADE WITH DEADLINE TBD BY COMMUNITY (STRAWMAN SUGGESTION MONDAY, MAY 29th, 19:00 UTC)

While preparing the cross-chain VRSC currency launch and as part of continual review, we determined that one numeric function that is used in the cross-chain challenges and is calculated differently for mainnet, due to the original launch that included the deprecated time locked rewards, was incorrect. This had never been hit, as it would have been calculated incorrectly only if multiple chains experienced a challenge of two competing forks, but in that case, it would falsely reject any challenge proof. That would mean cross-chain challenges on mainnet, even though they were fine and fully tested on testnet, could fail and block a bridge until that calculation was made correct and the network upgraded.

Because we have the notification oracle technology, we triggered a rip-cord notification selectively for cross-chain that made all nodes listening to the oracle, which we expect to be near 100%, disallow cross-chain operations without affecting same chain operations. That means that all functions on the Verus network, except launching PBaaS chains or Ethereum bridges are fully functional and unhampered in any way at this time. If someone tries to launch a PBaaS chain while this oracle notification is active, all nodes following the oracle will reject such an action without incident until the notification is removed.

This release, v1.0.5, properly calculates the aptly named “magic number” for each chain, in the correct way on the mainnet network, and as long as the notification is set, is fully compatible with current versions on the network.

Once everyone has had a chance to upgrade to v1.0.5, the network/community should agree to remove the notification from the oracle and allow PBaaS launches without false rejection of cross-chain challenges. Once we do that, and when someone launches a PBaaS chain that issues such a challenge, the network would become incompatible with Verus versions prior to 1.0.5.

From our perspective, we are not aware of urgent efforts to launch a PBaaS chain in the next few days by anyone who cannot wait. All things considered, we believe that we can at least wait until the community discussion on Saturday, and that a strawman deadline would be **Monday, May 29th, at 19:00 UTC** for everyone to have upgraded. If you are urgently wanting to launch a PBaaS chain, have a plan, and are prepared to do so sooner, we invite you to join the community meeting on Saturday at 19:00 UTC and let everyone know.

Having this notification remain in place until removed keeps the network running smoothly as it has been since activation, with all DeFi, multicurrency, and non-cross-chain functions enabled. Since the ETH bridge is undergoing final changes and review, is not quite ready to launch on testnet or mainnet, and will likely require at least a couple days on testnet to finish its review, we do not believe the current notification is hampering or slowing anything or anyone down at this time.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your

funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved...

Assets 6

1 person reacted

v1.0.4

Asherda
v1.0.4
`5ab050b`
v1.0.4

Announcing v1.0.4 - CRITICAL, NON-MANDATORY MAINNET UPGRADE
THIS UPDATE IS CRITICAL FOR MAINNET DEFI OR CROSSCHAIN WALLET FUNCTIONS, AS IT RECTIFIES POTENTIAL REFUND FAILURES IN EDGE CASES - IT IS NOT REQUIRED FOR MAINNET SYNCHRONIZATION OR MAINNET FUNCTION PAST PBAAS ACTIVATION, BUT IT IS HIGHLY RECOMMENDED.

DO BE SURE TO UPGRADE TO v1.0.3 OR LATER BEFORE MAY 23, 2023, EXPECTED BLOCK #2549420

FOR CONTINUED TESTNET USE, UPGRADE ASAP

Mainnet improvements:

- Properly inserts refund addresses in all cases when making DeFi or cross-chain transactions
- Merge mining improvements from @hellcatz for higher merge mining performance on all platforms and full pool merge mining support when used in combination with @hellcatz and @Oink70's SNomp pool improvements
- CLI now defaults to mainnet for all PBaaS chains unless -testnet is specified

- 2 versions of GUI will be released for now, a mainnet and separate testnet version. Full PBaaS support in the wallet will only be for either mainnet or testnet, depending on the version used.
- Extends deprecation height from the previous 20 weeks to 52 weeks

Testnet fixes/improvements:

- Separate GUI release for testnet
- Refund address fix

This the is last expected and recommended release before the mainnet activation, Tuesday at block #2549420, which will add some pool merge mining improvements. Please make sure you update to v1.0.4 or greater before the activation block.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the

blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh

private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, s...

Contributors

-
-

hellcatz and Oink70
Assets 6

1 person reacted

v1.0.3

Asherda
v1.0.3
42e83aa
v1.0.3

Announcing v1.0.3 - MANDATORY MAINNET UPGRADE - HOPEFULLY THE LAST IN A WHILE - ACTIVATION DATE HAS BEEN PUSHED BY 2 DAYS

FOR CONTINUED MAINNET USE, UPGRADE TO v1.0.3 OR LATER BEFORE MAY 23, 2023, EXPECTED BLOCK #2549420

FOR CONTINUED TESTNET USE, UPGRADE ASAP

Mainnet: fixes:

- Fixes an issue discovered that could have resulted in behavior different than what has been tested in an edge case when mainnet activates

Testnet fixes/improvements:

- Forgives issues pre-1.0.2 on testnet that were blocking challenge resolution by lacking complete cryptographic proof for cross-chain notarization confirmation

What’s New

v1.0.3 has extended the API fundrawtransaction to allow Verus Mobile and other applications to easily construct transactions for DeFi conversions and cross-chain operations as well.

We expect at least one more non-mandatory release before the mainnet activation, Tuesday at block #2549420, which will add some pool merge mining improvements. Please make sure you update to v1.0.3 or greater before the activation block.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user’s keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus

PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch

discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchain...

Assets 6

1 person reacted

v1.0.2

Asherda
v1.0.2
576c5b2
v1.0.2

Announcing v1.0.2 - MANDATORY MAINNET UPGRADE
UPGRADE TO v1.0.2 OR LATER IS MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

Mainnet: fixes:

- Fixes a regression that prevented staking on locked IDs as well as modifying locked IDs

Testnet fixes/improvements:

- Fixes issues discovered when challenges occurred on the Gravity chain that blocked resolution of cross-chain challenges.
- Addresses all known issues discovered in testing

What’s New

v1.0.2 has no feature changes, some minor API fixes, proof improvements, and we believe the third time's a charm 🙂. The testnet will transition to the v1.0.2 protocol with no reset required, Wednesday, May 17, 2023 0:00:00, UTC.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming

- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an or...

v1.0.1

Asherda
v1.0.1
`d733d8e`
v1.0.1

Announcing v1.0.1 - MANDATORY Mainnet Upgrade, UPGRADE TO v1.0.1 OR LATER IS MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

Issues addressed

In addition to fixing an issue discovered by cautionfun#3236 on the current public testnet, this version also addresses an issue discovered during additional coverage testing in the oracle based, reversible rip-cord implementations to ensure the ability for the network to respond to any unexpected events quickly, safely, and without compromise to decentralization.

What's New

In addition to the full PBaaS protocol and all capabilities previously described, version 1.0.1 will also include some new capabilities that were previously not in the PBaaS protocol. The capabilities basically extend the idea of a centralized currency to include the ability to launch temporarily centralized currencies, including those that can issue identities. This is achieved by setting the "endblock" on a centralized currency launch of either a token or liquidity basket, enabling:

1. Minting of currency on demand by the identity of the currency as part of the `sendcurrency`command.
2. Burning a liquidity basket currency, but instead of affecting the supply/reserve ratio as normal burns do, the identity of a centralized currency can also burn currency to affect the reserve ratio weights of the currency.
3. Registering identities on a currency. When an identity is registered on a centralized currency, the fee for the registration is not burned into the currency, but sent directly to the identity of the currency. This is true for token or liquidity basket currencies. For decentralized currencies, fees are burned, which on a liquidity basket, puts them into the liquidity pool for all LPs to share in earnings, and for a standard token, reduces the supply and available future registrations on that currency.
4. In v1.0.1, setting the endblock of a centralized currency causes all centralized capabilities of the same ID to end, converting it in whatever state it currently is in a decentralized currency with no special control over it after that occurs by any identity on the network.
5. v1.0.0 did not allow a decentralized non-liquidity token to register identities. v1.0.1 does.

Thanks to ejuliano#8606 for suggesting the initial idea that led to the realization that these features could be added quite easily and with very little protocol risk. All of these new capabilities have had full coverage testing and validation over the past week thanks to @Asherda's leadership on that front.

Meanwhile, @alexenglish, @monkins1010, @Asherda, quipacorn#5205, and others have been working on the Ethereum bridge deployment in preparation for this release. The testnet will transition to the v1.0.1 protocol with no reset required, Monday, May 8, 2023 4:00:00 PM, UTC.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operatio...

Contributors

-
-
-

alexenglish, Asherda, and monkins1010
Assets 6

1 person reacted

v1.0.0

Asherda
v1.0.0
`f672ccf`
v1.0.0

Announcing v1.0.0 - PBaaS Mainnet Upgrade, UPGRADE TO v1.0.0-x MANDATORY FOR CONTINUED MAINNET AND TESTNET USE AFTER MAY 21, 2023, EXPECTED BLOCK #2546600

What’s New

This release delivers on every part of the vision (and most stretch goals) either described in the vision paper of June 2018 or ever planned for the PBaaS release. Thanks to the incredible combined efforts of so many people in the community ranging from development, companies and projects joining and their open contributions (Valu/Arkeytyp, CHIPS, vDEX, VaultAlert, cragslist, and more) to community members supporting users to helping educate others who can contribute as well, we have run these protocols now for years as we’ve improved them, and all of these capabilities will go live on the Verus mainnet May 21, 2023, with the activation expected targeting block 2546600.

As a community, we’ve discussed what the Verus PBaaS protocol can do for years. Now that it has existed on testnet for as long as it has, only those who have either been part of those conversations or experienced it first hand have a real sense for how much better Verus PBaaS is as a solution for cross-chain, DeFi AMMs, decentralized markets, self-sovereign provable recoverable IDs, NFT capabilities, anti-phishing, anti-MEV, scale, or just about any of the challenges people have on crypto platforms today. In fact, as we prepare for activation of this protocol that enables so many new use cases for crypto, not to mention easy onboarding when entrepreneurs discover how to really leverage VerusIDs, PBaaS is so far beyond what people are experiencing on any crypto platform today that it has been easy to dismiss our community as describing the impossible.

Once we are live on mainnet and people can actually use all of this themselves, along with every project, chain and currency on the PBaaS network and EVERY ERC20 or ERC721 on Ethereum or even bridged to Ethereum, the truth of Verus will be self-evident, and those who build on Verus will more easily build faster, better, more secure apps and services with the possibility of provable identity + privacy, crowdfunded projects, businesses, economies, and public infrastructure efforts all seamlessly integrated into UIs that do not need business deals, permission, or any centralized infrastructure to connect services and users, enabling everyone to communicate in provable, private, or public ways that always include bidirectional, secure commerce of all kinds.

This core capabilities of this release include (lots to learn to understand it all, but here’s a partial list):

1. **Mainnet PBaaS Activation MAY 21, 2023, EXPECTED BLOCK #2546600**: v1.0.0 is MANDATORY, meaning that you will need to upgrade to this release or a later version v1.0.0-x, before MAY 21, 2023 to stay reliably connected to mainnet. Testnet will also be reset within the next day, and this release will allow you to connect to the new testnet, which we see as a core development platform and staging area for the Verus PBaaS network and have no plans to reset in any foreseeable future.
2. **Verus Intersystem Protocol (VIP) - Layer 0 Multichain and Inter-chain Protocol**: To our knowledge, the VIP protocol is the only fully decentralized, provable cross-chain technology available on any network that is based on cryptographic proofs of each chain with optional witnesses to confirm chain state. From what we’ve been able to learn about LayerZero, Cosmos, Polkadot, Thorchain or others, VIP is different from (though closest in some ways to LayerZero), superior to, and more decentralized than all other cross-chain protocols we have seen. We can of course get into any level of discussion on the topic, but expect more descriptions and educational material to follow, now that the protocol was originally conceived of 5 years ago, and has been developed and tested/hardened for years.

3. **VerusID Content Multimaps - Unlimited Provable Data for Every VerusID**: With the PBaaS upgrade, VerusIDs gain a powerful new capability that can be used for social networking, voting, oracles, publishing any information, various forms of provable attestations, ratings, and much, much more. With the scalability of PBaaS and VerusID content multimaps together, the Verus PBaaS network becomes the most scalable, permissionless, provable source of human data from which AIs can learn about humanity in the world. We don't believe anything else even comes close, and with the recent rise of AI, it feels great to have this come together as a core original goal at a very appropriate and important time in history.
4. **Permissionless Chain, Token, and Liquidity Basket Currency Launches**: Whether you are looking to crowdfund an effort in a manner much like Kickstarter or IndieGoGo, sell identities to which you expect to add value or that are sub-IDs of a very cool root ID, or launch an entirely new blockchain economy and blockchain that starts with all of the Verus technology and a bridge to Verus and Ethereum from day 1, Verus is probably the best platform for you. In fact, as we were planning for the next testnet and wanted to have our own version of DAI to use in the bridge, since we want only decentralized currencies in the live mainnet bridge, we used a command to launch a token on the old testnet, exported it and all its supply to Ethereum's Goerli on the existing bridge, and that will be the DAI proxy/simulant that we will use on the new testnet. It saved us all some time, proving that Verus PBaaS will also be the easiest, most efficient way to launch even an Ethereum ERC20, whether that ERC20 represents a token, DeFi basket, or even another PBaaS blockchain 🙂
5. **Decentralized Ethereum Bridge**: Shortly after PBaaS activates on mainnet, the Ethereum bridge will launch as a decentralized gateway and 1:1 provably mapped currency called "vETH" on the Verus network, also available to all PBaaS chains. The Ethereum bridge will also include a 100% backed basket of 33% Verus, 33% Ethereum, and 33% DAI, which will have the following functions:
   Auto conversion of fees from Verus <-> Ethereum when sending cross chain, based on the on-chain conversion price in the liquidity basket
   Permissionless ability to register *.vETH IDs, which in addition to the normal sub-ID capability of creating a single token that has control over the ID and can be exported to Ethereum as an automatic ERC721 NFT, can alternately be used to create a "mapped currency", which can be provably mapped 1:1 to any ERC20 currency on the Ethereum network by exporting it to the vETH bridge. Only Verus root IDs, and IDs of a gateway can create mapped currencies to a gateway.
   Decentralized, fair bridge launch. The vETH bridge will launch as other currencies, chains, and gateways can launch, including the bridge converter. When the bridge launches, the contracts will have a surplus of Verus that comes from the fees paid to launch the gateway. In the protocol, that surplus is first used to solve the chicken and egg problem of none of one currency on the other chain. The contracts will cover the Verus fees for all sends from Ethereum before the Bridge converter launches, the liquidity basket becomes active, and the bridge recognizes that launch. Once that happens, the remainder of ½ the fees (5000 VRSC) that are left over from the launch and did not go to miners and stakers or pay Verus fees for people sending from Ethereum

6. **The Verus Fee Pool and Rewards**: Another technology and solution unique to Verus, the Verus Fee Pool technology goes live with PBaaS. What this means is that as people use the cross-chain, DeFi, or purchase/register IDs on the network, the miner or staker will put all fees into the on-chain “fee pool”, and then take 1/100th of the pool in addition to the block reward. This gives everyone incentive to still prioritize transactions based on fees, while preventing validators from gaming the fee system by washing fees-in vs. fees-out or working to reorg/rewind to capture a particularly juicy block reward. Instead, this technology aligns all validator incentives with the health/proper operation of the network and creates a circular on-chain economy that can last well beyond block rewards.

For those who actually understand the challenges actual users have in crypto today, this release represents a historic move forward in public cryptographic networks and credibly and actually neutral infrastructure for society-wide human and AI collaboration and learning. As a community, this version really does realize the vision laid out in the original Verus Vision Paper, and at the same time provides much more than was described there. We are a community of individuals, and it is only because that is what we are that we have all arrived here together. The rest of the world is stuck on their Munchausen’s protocols, trying to figure out how to share the front-running and back-running spoils taken from users, but Verus activation gives them a better path forward.

As always, we do not promise, we describe a vision and welcome contributors from everywhere. Now that we, as a community, have delivered on the original phases of the vision together, it is time for all of us to consider what we might want to build over this incredible, geoscale, ID-enabled, fully decentralized platform. Our next efforts across the community should be to realize the promise of this network in the use cases we create, use cases that we will have an advantage creating on Verus over any other platform because it really is that much better.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertisin...

Assets 6

2 people reacted

v0.9.9-5

Asherda
v0.9.9-5
`a7adac2`
v0.9.9-5

Announcing v0.9.9-5 - CRITICAL PRIVACY FIX PARSING FRIENDLY PRIVATE NAMES, CRITICAL UPDATE FOR RELIABLE TESTNET OPERATION

Mainnet Changes - Wallet Hardening

v0.9.9-5 Fixes an issue introduced in v0.9.9 during the ID name hardening work that prevented parsing id@:private to the z-address of id@ when running a native node. Instead, such an address parsed to the main id@. Since this issue could impact on-chain privacy of someone intending to send to or from a private address and using friendly names, we are considering this a critical privacy release.

This version also prevents the wallet from crediting transactions that can be constructed to make a burn look like a spendable output, ensuring that unspendable funds are not added to a wallet or address balance. @Alrighttt, who has been focusing on Verus security testing, reported a way that someone might create such an output, and we appreciate that it is before we have ever seen such transactions on the public or test networks. Thanks @Alrighttt for the report and discretion that helps keep them off the network(s) altogether! We recommend that everyone update as soon as possible to version 0.9.9-5 or later.

We are working towards the mainnet PBaaS upgrade version 1.0.0, but we cannot put a specific date beyond ➡️SOON on its release.

What's New for TestNet

This release fixes hopefully the last remaining issue that was causing testnet users to experience sync issues on PBaaS chains.

There are no other significant changes.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that incre...

Contributors

-

Alrighttt
Assets 6

1 person reacted

v0.9.9-4

Asherda
v0.9.9-4
`94df9fe`
v0.9.9-4

Announcing v0.9.9-4 - RECOMMENDED FOR MAINNET, MANDATORY UPDATE FOR RELIABLE TESTNET OPERATION

Mainnet Changes

Low level stakeguard improvements, no functional changes

What’s New for TestNet

This release fixes an issue that was causing testnet users to experience sync issues on chains that had more than one oracle upgrade for the same capability activated.

There are no other significant changes.

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don’t worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It’s the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your

funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all currency conversions that use a particular liquidity pool and are mined into one block are solved and priced simultaneously, addressing the problems of miner extracted value (MEV) and front-running, while providing fee-based DeFi integrated incentives to miners and stakers, ensuring smooth consensus operation and fee conversion capabilities by integrating DeFi liquidity pools directly into the consensus and cross-chain bridge protocols.

Simultaneous blockchain and blockchain liquidity pool launches

- Launch of a world class, worldwide, merge-mineable blockchain along with a fully decentralized or centralized “bridge” converter liquidity pool as part of defining a new blockchain. Bridge converter currencies have the same flexibility as other fractional 100% asset backed or partially asset backed currencies, but is bound to the launch of the new blockchain, runs on the new blockchain, and all fees generated via cross chain fee conversions or general use of the liquidity pool are earned on the new blockchain with no rent going back to the Verus blockchain, only seamless connectivity.

Blockchain-based, crowdfunding currency launches with minimum participation or automatic refunds, including for dual launches (blockchain and bridge)

- Set required minimum levels of worldwide participation in your preferred currencies on chain. If by the start time of your blockchain, minimums are not met, all participants will automatically get a refund of all of their pre-conversions, less the network fees. The launch options also provide for maximum participation in one or more currencies, pre-launch discounts, price neutral pre-allocations to select IDs that increase the fractional reserve ratio to issue currencies, similarly price neutral carve-outs of proceeds, and pre-launch discounts

for early participants. Using VerusIDs, launches can also include vesting schedules in the pre-allocations as well.

An interoperable, multichain network for new use cases and unlimited scale**

- The Verus multi-currency, multi-chain network allows the creation of an unlimited number of interoperable blockchains in the Verus network. Notary IDs, specified at chain definition, provide decentralized blockchain-specific bridge confirmation, enabling public blockchains available to the world for merge mining and staking, as well as private, internal blockchains, which are easy to setup with easy bridging of public currencies into an organization and onto their internal private network and back, with all features and currencies of the public chain but none of the access. There is no limit on the number of blockchains that may continuously operate and interoperate on the Verus network. While there is some overhead for cross notarization, the model for the...

Assets 6

v0.9.9-3

Asherda
v0.9.9-3
`2213c9f`
v0.9.9-3

Announcing v0.9.9-3 - FULLY OPTIONAL UPDATE FOR MAINNET, MANDATORY UPDATE FOR CONTINUED TESTNET OPERATION

Mainnet Changes - no protocol changes

V0.9.9-3 introduces the `prunespentwallettransactions` CLI command and RPC API, providing a method for holders on testnet or mainnet of very large wallets to prune older spent transactions, which can improve performance for wallets that may be slowing down due to size. Thanks @himu007 for the PR! There are no mainnet protocol changes in v0.9.9-3 over the prior release.

What’s New for TestNet

This release includes an activation that allows a major upgrade to the testnet Ethereum bridge contracts that is both not expected to be needed again on testnet and not ever needed on mainnet with the new contract upgrade model. This upgrade should clear all known issues on the bridge to Goerli, meaning any remaining bridge transactions will be delivered, and the ETH bridge should then be fully functional.

The Path for PBaaS to Mainnet

This testnet has actually been through a lot already, more than we expected before mainnet, and this last experience, along with its full restoration of cross-chain and all function has been very useful and was the impetus for the Verus Oracle Notification technology. In the process of getting here, we implemented the necessary changes and fixes to get everything back on track, but we also have remnants of these events in the daemon, and on the final protocol, these are there to enable compatibility with the history of testnet, not because we need these upgrades on mainnet PBaaS. It gave us experience, increased confidence, and pushed us to develop a fast, decentralized network response capability. At the same time, it would be best for our code base, both on the Verus side and in the ETH contracts if that testnet history is removed from the code before mainnet activation. Doing so would make the next upgrade incompatible with today's testnet and its earlier history.

As a result, we have decided that we will create one final testnet, but this time, we would do it a little differently than in the past. We will extend the pre-PBaaS upgrade on the existing testnet to a future date via the oracles. If you have started a chain and control the default oracle on any chain, please check the #pbaas-development channel for instructions on setting your oracle to extend the upgrade.

We will leave the current testnet running and fully functional, as we prepare the next release, which we expect to be ready for a mainnet PBaaS activation. We will effectively abandon this testnet and leave it running long enough to overlap with the next and final testnet instance before PBaaS release. You should be able to actually send currencies over from the first testnet to the second via Ethereum's Goerli testnet, even map some VRSCTEST-CLASSIC or something else through the bridge, but we do not intend to update that daemon again, or continue to run nodes or notaries on the old test network for more than a couple weeks after the new testnet goes live.

Thanks for all your help and contributions, both already made and those to come for this historic upcoming release!

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's

the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today’s decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. In doing so, they will start with a complete, multi-currency, ID-capable blockchain with DeFi capabilities that is merge-mineable and stakeable with other blockchains in the Verus network.

Consensus integrated DeFi liquidity pools and fractional currency baskets

- Any ID owner may define Verus DeFi fractional basket currencies with one or more asset currencies backing the liquidity pool at a fractional percentage ranging from 5% to 100% backing. The Verus DeFi protocol ensures that all curren...

Contributors

-

himu007
Assets 6

v0.9.9-2

Asherda
v0.9.9-2
69628ef
v0.9.9-2

Announcing v0.9.9-2 - CRITICAL SECURITY UPDATE FOR MAINNET AND REQUIRED UPDATE TO RESUME SUCCESSFUL MERGE MINING NOTARIZATION ON TESTNET

Important Mainnet Security Updates**

v0.9.9-2 introduces new mitigation code for the **Rab13s** vulnerability, announced by Halbourne 3 days ago as a zero day vulnerability in over 280 blockchains, including Zcash and Verus, that could allow someone to attack and bring down specific nodes that they may target through exhausting the node's resources. The Verus fix adapts a fix implemented in Zcash that also does a better job of preserving privacy while gossiping transactions around the network by making it harder to track their node of origin via listening nodes.

This release also includes an important update of the OpenSSL library, which Verus uses for encrypted node connections and addresses a buffer overflow vulnerability in that library.

What's New for TestNet

This release re-enables cross-chain notarization, and represents the completion of a great deal of learning from the first roll out of auto notarization. Basically, we encountered two issues:

1. There was a notarization bug in a significantly updated approach that caused notarizations to fork, effectively simulating a worst case, unmitigated attack on the notarization system. While that did not cause the system to fail per se, it resulted in the generation of many perceived, although not real, notarization forks on the network.
2. Due to our effort to err on the side of more evidence than we believed was sufficient with auto notarization, then optimize to reduce the proofs required, the protocol was simply taking too long and generating too large proofs to be reasonable, required, or what we need on mainnet in any real scenario.

As a result, we will take a few days to change some parameters in the auto notarization protocol that will both reduce the total and maximum amount of proof required, as well as provide a sufficient level of cryptographic proof, similar in strength to the current proof model with more efficient and selective proof choices. We don't expect this important step to take long, and until then, we will leave testnet on the pre-PBaaS to mainnet preparatory state.

The good news is that since we have the new Verus Notification Oracle technology, we can enable everyone running on the current testnet to remain connected as all cross-chain transactions to and from PBaaS chains will resume flowing. We are still seeing an error in BridgeKeeper, but since this release includes a mainnet security update and addresses all other known issues on testnet, we'll release this immediately and update Bridgekeeper as soon as possible. In order to merge mine PBaaS chains with Verus or the Ethereum bridge to Goerli going forward, you will need to update to v0.9.9-2, which everyone should run for mainnet anyhow.

Once you update, please feel free to resume merge mining all PBaaS chains and Verus. You will need to wait for the BridgeKeeper update to be able to merge mine and stake with Ethereum's Goerli.

Thanks for all your testing! We've learned a great deal from this upgrade, and although it may feel like a delay that we took the time to learn and address it in place on the public network before moving on, this learning and consequent hardening will ensure the mainnet release will be more hardened than ever. Next steps are to upgrade auto notarization to be resilient to anything like this level of buildup over weeks automatically and prepare the mainnet release. Join us in testing everything in the #pbaas-development channel. Contribute as you learn and build, and let's all make history together!

Additional Verus Capabilities

- On-chain Launches of Token, Centralized Currency, and Liquidity Basket AMMs
- On-chain Launches and Merge Mining of Independent, Connected, Interoperable Blockchains without Programming
- On-chain Self Sovereign, Provable Identities, NFTs, and Individual or Organizational Profiles

Verus ID and NFT Marketplace

Buy and sell VerusIDs on-chain, advertising your offer directly to the owner of an ID or NFT, or posting the sale of your NFT on the worldwide blockchain for all the world to see. Execute transactions in a completely decentralized way. Pay or offer to pay from a transparent or zero-knowledge private address, still auditable by you. Accept payment to either as well, and best of all, execute your transactions directly, peer-to-peer without any intermediary necessary. Don't worry the on-chain model still makes room for owners to select and share proceeds with value added agents, marketing organizations, or other participants in a new economy of provable digital ownership. It's the next step in the evolution of VerusID, the most powerful self-sovereign identity and secure storage model for funds in the digital world.

Verus Vault

With Verus Vault you can now protect funds on a VerusID, even from theft of a private key! If you lock your VerusID with Vault you cannot spend funds from that identity at all until it is again unlocked. While locked, you can still stake those same funds on the Verus network and earn by doing so. Of course, you can also still receive funds.

IT IS IMPORTANT TO NOTE THAT ENABLING REVOCATION, RECOVERY, AND ALL VERUS VAULT CAPABILITIES REQUIRE YOU TO HAVE ONE PRIMARY IDENTITY, AND AT LEAST ONE REVOCATION/RECOVERY ID CONFIGURED.

A locked VerusID can always be revoked and recovered by its revocation and recovery authority identities, which circumvents the lock. At the same time, anyone with only the primary keys, even a multisig of primary keys must first unlock, then wait for the predetermined unlock time before they can spend or access funds. This gives you, or maybe a company that specializes in watching the blockchain to whom you've assigned the revocation ID to revoke and recover whenever an

unauthorized unlock occurs. That means that like a bank, setting a 24 hour unlock delay on your locked IDs actually provides the first decentralized solution to the infamous 5 dollar wrench attack.

In addition to a new level of blockchain protection and decentralized funds recovery, Verus Vault provides the same security for your IDs and NFTs as well as time locks for other purposes, such as vesting schedules, trusts, and inheritance. With Verus Vault, you can now protect and recover your funds, preserving all your assets and generational blockchain wealth from common forms of crypto loss or theft, no bank required.

New Verus Multicurrency, Multichain, DeFi Enabled Testnet

With an easy GUI for basic operations or command line for more advanced functions. Without any programming, you can now create new identities, currencies, liquidity pools, and blockchains for your business, your government, your projects, a worthy cause, your family, or your next decentralized application suite. Send currencies worldwide on the same chain, or across blockchains with ease. Even convert currencies to others on the network without an exchange by sending to yourself and converting along the way.

The new Verus testnet is a full-featured, intrinsically decentralized multi-chain blockchain platform with an unlimited number of identities, currencies, liquidity pools, and blockchains. It is accessible from the released versions of Verus Desktop and Verus CLI wallets, and it is the beginning of a new age in crypto. There are so many things you can do with Verus that you cannot with any other cryptocurrency platform, and you can try them all today.

As Verus PBaaS offers completely new capabilities that go beyond today's decentralized platforms in many fundamental ways, the worldwide Verus community put its energy into creation, rather than convincing everyone that its capabilities are possible. Members across the Verus worldwide community have worked hard to make this all possible, and we are more than excited that you can now experience it firsthand. If you have an interest in the future of crypto, you owe it to yourself to learn about Verus, an unlimited scale, decentralized future with truth and privacy for all.

The Verus testnet, available in the Verus Desktop or cli wallets as the VRSCTEST coin, has the following capabilities, which to our knowledge are unique in crypto today.

Self sovereign, revocable, recoverable identities (currently on mainnet) VerusID

- Enables permissionless registration of friendly name strong identities and funds addresses that are simultaneously fully self-sovereign, revocable, and recoverable.

Staking-capable time locking and theft prevention (Verus Vault)

- Enables identities to be locked, preventing any funds under their control from being spent while locked, but still allowing seamless staking of funds. When locked, a user specifies an unlock delay, typically long enough to notice when someone who might have compromised a user's keys would have to unlock the ID before spending. The only way to circumvent the unlock delay is to revoke and recover an ID. Users may also choose to create and use fresh private keys when unlocking an ID as well. This enables virtually theft proof workflow and

a solution to inheritance, trusts, vesting schedules, the 5$ wrench attack, and identity theft. IDs may be used as friendly name cryptocurrency addresses for all currencies on all Verus PBaaS blockchains in the Verus network. The VerusID protocol is a protocol, which can also be implemented on non-Verus systems.

Multi-currency, user created, decentralized tokens and merge-mineable, interoperable blockchains without programming

- Enables any user with an ID to create their own token currency or even full fledged, multi-currency, ID-issuing 50% POW/50% POS, 51% hash attack resistant blockchain that can send and receive from the Verus chain which launched it. All PBaaS chains run from the same daemon, and projects may choose to join the worldwide Verus community in improving the daemon. I...

Assets 6

Footer

© 2023 GitHub, Inc.

Footer navigation

- Terms
- Privacy
- Security
- Status
- Docs
- Contact GitHub
- Pricing
- API
- Training
- Blog
- About

Releases · VerusCoin/VerusCoin · GitHub

Privacy, Community, Power

Verus consensus non emitur

“True consensus cannot be bought” - anonymous

Michael J. Toutonghi Michael F. Toutonghi Alex R. English

June 12, 2018

June 12, 2018

Abstract

The Verus Project aims to establish a secure, privacy-centric, fairly-distributed cryptocurrency. But – beyond this currency – Verus seeks to become much more than a zero-knowledge privacy coin, one with two completely new highly-decentralizing proof of work and proof of stake algorithms.

In addition to payments, decentralization, and privacy features, Verus Project plans include its direct use as a currency for provisioning scalable and secure public blockchains as a service (PBaaS), for Verus applications built upon these parallel chains to scale. What this will do is simple: It will enable all people – as well as all nodes on the Verus network to participate in and benefit from a decentralized, blockchain service economy.

This paper details the Verus vision and describes the function of Verus as its own platform, and also as a member of the Komodo platform ecosystem, in the context of its first applications. Verus core applications will provide a foundation to build additional applications and services, which will leverage Verus' automatically created blockchains, called autochains. Autochains – or PBaaS – will be provisioned and notarized by the Verus blockchain miners and stakers, in exchange for Verus currency.

Autochains will be validated through proof-of-stake by their user populations. In addition to extremely scalable, dynamic, publicly-secured autochain applications, this will add a dynamic isolation and security component to applications that can also create, manage, and verify transactions on the main Verus chain or any other Komodo-compatible, Crypto-Conditions [17], Interledger Protocol [25] enabled blockchains.

The ways we apply this technology to our world – to our biggest contemporary challenges – has the potential to completely remake the fabric of our society.

Preface

Human progress leapt forward with the invention of money. Money enabled worldwide, trade-based economies to move from a primitive, barter-based system to a consensus-based valuation and accounting of verifiable, storable, divisible, and scarce commodities as early currencies of exchange.

Even in recent times, actual or perceived scarcity and authenticity of source, whether genuine gold, an original giant coin of Micronesia [27] – or government backed notes with trusted fiscal management and the ability to exchange for oil – provide the foundation of value upon which, ultimately, human resources are bought and sold.

What blockchain does is straightforward. It enables the creation of cryptographic tools and applications that do nothing less than provide us with a new future of programmable money. This programmable money integrates directly with existent accounting functions and, eventually, with all automated services. Blockchain technologies provide humans with a new, verifiable solution to ensure scarcity, authenticity of source, inbuilt transferability, fungibility through privacy, and programmable rules. More than that, fully peer to peer, public blockchains disintermediate and provide tools that offer the opportunity to revitalize our economic systems and societies, potentially creating a more equitable and honest economic framework for all.

About This Document

This vision paper is not intended as a technical reference, but as a vehicle to communicate our vision, our plans, and our thinking – as we work on realizing the true potential of Verus.

All technologies described are based on a clear understanding of methods intended to implement each solution but – as with any project – the specific implementation details will be refined as we actually realize their functionality. To supplement this vision paper, we will follow soon after its initial release with a white paper, describing the new technologies developed for Verus Coin's initial launch, what benefits they provide, and provide details of their implementation. We will continue to supplement this vision paper with white papers, as appropriate, for each completed phase of Verus Coin's development, in the future.

As of this point, we have completed Phase 1, having released Verus Coin with zero knowledge privacy, two brand new algorithms combined for simultaneous CPU-mineable proof of work (POW) and fair proof of stake (POS) consensus, in-wallet mining and staking on PCs, leverage of and support for advanced Komodo platform technologies, and wallets and miners for every major PC operating system.

1 Introduction

Information vies for our attention in today's digital world, trying to convince us of what product to use or which politician to believe. We are watched and measured as we react to that information, in order to convince us what to buy or to believe. This is an active process that exploits our lack of online privacy – combined with weaknesses in human psychology – in order to make profit; extracting value from, even affecting our behavior, learning about and influencing each of us individually to open our wallets or add our voices to another's agenda.

No single person can digest and fully understand – let alone verify – a fraction of the information thrust at us in daily life. We are told to give up on the notion of privacy and to trust networks of companies with our most private data, our identity, our credit histories, our location, our habits. We are also told that our voices are lost in the digital sea of information. Yet how is this the same sea through which we are laser-targeted based on correlating our behaviors to learn so much about us, individually? What if these technologies could be turned towards the benefit of society, first – and then allowed to support businesses, in that context?

We on the Verus team believe it **is** possible to support businesses and governments requirements with digital systems that: 1) respect your privacy, 2) give you control over your data, and 3) enable you to speak your mind with the anonymous authority of an authorized voter or member of a community, in a way that can directly be heard and affect actual change.

Before we explain how, it would help to understand the shared beliefs behind our vision for Verus. These beliefs underpin everything we build into the Verus network:

1. We believe that every human has ideas, knowledge, and value to contribute to our society. By using technology to reward people for their contributions, we can enable each of our verifiable, yet anonymous voices to be heard as a collective truth.

2. We believe that those who contribute positively over time to the system should be rewarded for that contribution and provided with ongoing incentive.

3. We believe that a world-scale, peer-to-peer system that can enable humans to be queried directly, confidentially, verifiably – and in a transparent manner – can directly provide populations of people and the world with valuable, transformative tools.

With Verus, we will introduce digital tools to enable us all to build a better world together. We will monetarily incentivize – with our technologies – behavior that strengthens communities and institutions. This is the thing that's missing in the online world today: Fiscal incentive for communal behavior built into the very fabric of its function.

The Verus Project's tools will make it easy to create an identity – or multiple identities – on the Verus multi-blockchain system, which can accumulate value and even have multiple personas, each to represent a facet of your identity as a whole. This reflects how we might express identity in our personal or professional lives, where some situations call for provable credibility, yet others require no more information than what you might reveal when encountering a casual stranger.

Each identity will have its own, unique address. Unless its owner reveals information to link two or more identities, it is cryptographically hard – meaning virtually impossible – to correlate one identity to another. At the same time, the owner of an identity can still cryptographically verify statements made about identities under their control, attesting to identity properties (such as passport, age, height, citizenship, photo, etc.) as strongly as is possible with today's digital technology. As part of the Verus vision, to be described in more detail in later phases, we intend to support fully decentralized verification of identities that can provide as strong verification as today's centralized systems. At the same time, we believe it is important for practical reasons to enable compatibility with centralized forms of identity and to enable people to optionally support KYC in identities. To enable a smooth bridge between centralized and decentralized identity systems, today's ID systems, including biometric and government issued IDs, will be supported via centralized or decentralized verification to enable use of Verus identity in situations that require conformance to know-your-customer (KYC) regulations. Theseforms of ID, however, are not required to establish or use even strong, decentralized identities on the Verus network.

Verus autochains, will operate parallel to the main Verus chain and enable large-scale applications – such as polls or elections – to run simultaneously without concern for congestion or excessive fees. Autochains will enable poll application users to provision their own secure blockchains just by using the application – spawning dynamic parallel chains that can process thousands (or potentially millions) of transactions per second when needed. Autochains will operate by proof of stake, enabling each chain to have security isolated to its direct user population. Autochains will also be backed by Verus notarization and block time synchronization – providing the full weight of its PoW/PoS security layer as well as the Komodo platform's delayed proof of work (dPoW), to provide notarization all the way back to the full power of the actual Bitcoin blockchain.

For **Phase II**, what we expect to be an extended development phase, we will work to implement autochains and their first application in the world. We will eventually use them to create, secure, scale, and perform polls for everything from classifying online content, to identifying real public opinion, to actual, real-time elections for an organization or – conceivably – a government.

Our goal is to make these polls easy to use from a PC or mobile device, yet industrial strength and suitable for serious, secure elections. They will leverage the latest cryptographic technologies for privacy – known as zero-knowledge proofs – to preserve confidentiality.

They will be:

- **Confidential** — No one but the voter knows who or what they voted for – unless the voter discloses. Results of the vote can be withheld until the vote is complete, at which time they can be released to everyone, simultaneously.

- **Verifiable** — Only voters that are authorized to vote can vote. Each voter can vote only once. Each voter can look at the released results and see that their vote was counted.

- **Transparent** — Anyone can validate the number of votes counted, and the number of votes,each person or selection received.

- **Secure** — Our novel autochains – dynamic, security isolated, proof-of-stake (PoS) blockchains – are layered over proof-of-work (PoW) and delayed proof-of-work (dPoW) security, all the way back to the real Bitcoin blockchain through Komodo.

  By default, the first layer of security is run by the actual voters, themselves. Together, this provides unprecedented layered security both in the autochain and in delayed proof-of-work, leveraging the network with the most hash power in the world.

  We like to think that – on a Verus autochain – a 51% attack is called winning a poll.

- **Scalable** — Each poll is conducted on its own, automatically-created-and-validated blockchain, operating under its own visible validation rules.

2 Our Vision for Verus Foundational Applications

Throughout history, understanding what a population truly thinks or feels has been an invaluable capability. The manipulation of that understanding can, and has, repeatedly changed the course of history.

The Verus Project plans to use highly decentralized blockchain technology to enable all people to safely, anonymously, and confidently express their opinions – on any issue. Verus users will be able to share their knowledge in a public forum, query a population of humans (or eventually both AIs and humans), quickly and effectively, and – importantly – earn cryptocurrency in exchange for these contributions to collective knowledge.

A primary goal of the Verus Project is to enable societies and organizations to make decisions based on a more honest understanding of the public's actual beliefs. To achieve this goal, we consider the need for both confidentiality and transparency, to ensure that – first and foremost – the system can be trusted. It is also critical to prioritize decentralization and develop a core platform that can leverage fully transparent blockchain technologies, while running many applications that respect privacy. With Verus, these applications can leverage zero-knowledge succinct non-interactive arguments of knowledge – or zk-SNARKs – the most reliable and tested iteration of proven, zero-knowledge privacy technology available today.

Our novel approach ensures that this technology can be used at scale, across any population size, throughout the world. Using the tested-at-scale infrastructure and tools of the Komodo platform, and Verus planned development of PBaaS – the Verus developers will build human organizational tools, including voting and fully self-sovereign identity with a reputation system that respects privacy. The goal, here, is a significant one, and these tools are planned in a way to allow entire human populations to transparently and directly share their knowledge and opinions – without the risk of privacy violation, spin or censorship.

Verus tools will also leverage human (and eventually non-human) intelligence at scale to solve previously challenging problems in a decentralized manner – financially rewarding those who contribute to their proper function.

To serve all people in the world simultaneously with these public, peer-to-peer tools, Verus introduces autochains. Autochains are a novel scaling model for blockchain systems. Autochains will enable full node miners to provide public blockchain provisioning services. These services will occur in transient, parallel chains, chains that are isolated from congestion, disruption, or interference from the main Verus blockchain. As an autochain operates, it is

almost completely parallel and isolated from the main Verus chain, except for the capability of posting transactions and proofs to Verus, for results and coordination across applications or people.

By combining privacy, polls at scale, and identity, Verus can be used to post polls and ask almost any question of a population or subpopulation of people, receive an honest – and, importantly – verifiable answer, and still respect the privacy of the respondents. When even the first support for polling is active on the Verus network, the Verus community will be able to vote on the long term direction of the Verus Project, itself.

Over time, Verus polls – used for purposes such as determining the accuracy of news online, for research, or even for political polling, will be combined with machine learning to enable all of us, as a society, to benefit directly from our collective intelligence without today's risks to our individual privacy. Verus will combine auditability and verifiability of transactions with the option of verified, confidential participation – meaning societies and organizations can directly understand what people think, feel, and believe, as populations – without taking control of their personal data or targeting some individuals, based on their answers or beliefs.

Ultimately, our vision for the Verus Project is to enable us all to directly participate in our own worldwide economy. Verus applications will enable any individual to speak up with the power of an authorized participant – and with the confidence of anonymity – to produce a verifiable, transparent, and honest flow of information throughout the globe. It will transform the way that anonymity functions in our current technological environment – prioritizing cooperation and providing the financial incentive for that cooperation to occur.

3 Important Verus Concepts

The longer term Verus Project vision of a better society through public blockchain services relies upon three fundamental pillars –where the correct operation of each strengthens the others in the form of a virtuous circle.

These pillars are:

1. Extreme transaction throughput and scalable decentralized applications with autochains, which provide public blockchains as a service (PBaaS)

2. Selectively strong, private identity, and

3. Open application support with a foundation of polls, voting, and lottery selection services.

Each of these pillars will help us process human knowledge, understanding, and/or opinion on any content or topic. Together, they will enable direct querying – with confidential and truthful – answers across an entire population. It will create a secure, public platform that respects privacy, and can potentially serve as the foundation of a more respectful society.

It is also important to know about the the concept of **Verus Virtue** when reading further in this document. The best way to think about Virtue for the purposes of this document is as a separate currency that can be earned, but not purchased or sold, and is part of the

evolution of the Verus proof-of-stake algorithm, slated for later phases of the Verus project. Proof of Virtue is a technology that will be described in much greater detail and implemented in later phases of the Verus project.

3.1 Autochains

With Verus, we plan to create a platform upon which we will build voting and even election systems that support our vision of enabling a better society through blockchain.

To run simultaneous polls across disparate populations on one blockchain-based system, we need another dimension of information – one that retains privacy, and is anchored to a primary store of value, Verus.

One approach to creating another dimension of meaning would be a system of "coloring." The Ethereum blockchain system and some layers over Bitcoin or Bitcoin compatible coins use a version of a coloring system. [5] [6] [7] [8]. While, at first glance, this seems like a reasonable approach, it is actually a suboptimal solution, at best.

With Ethereum and the ICO experiences of 2017 [9] [10], we now know what happens when you create one blockchain and overload it with transactions: Congestion and unnecessarily high fees. Developers have proposed many solutions to this problem of scaling – including a move to proof-of-stake systems and sharding (effectively trying to separate the various

functions on a blockchain), or even a move to large systems of parallel chains with a common design.

Yet instead of implementing a "solution" with colored coins sharing one blockchain, we have decided to take a completely new approach to this problem. Our approach will not only provide an extra dimension of information and an unlimited number of token types, or "colors", but it will do so with highly scalable PBaaS autochains, which also reward Verus miners and node operators as provisioning agents. This is a novel method of scaling, one which provides significant scale and security benefits through isolation of transaction processing for each application instance on its own blockchain.

Rather than thinking of Verus as a single blockchain, it is more appropriate to think of it as a blockchain-based system, one that is rooted in a primary value chain.

Autochains are exactly what they sound like – automatically instantiated blockchains that are validated initially – and when notarizing – by Verus miners, and otherwise, only by their users, often a disjoin set from the whole of Verus users. Autochains, once instantiated, can be used indefinitely or can have a finite lifetime to be used for a specific purpose. In this document, we will often discuss autochains in the context of a few concrete applications that we intend to implement on the Verus network. It is important to keep in mind that their versatility is **basically unlimited**. Due to their security being inherited from the main Verus chain, autochains eliminate perhaps the biggest disadvantage that creating consensus-based distributed ledger applications suffers from: Weak defense against attacks in their earliest stages.

In **Phase II**, Verus will define an autochain provisioning protocol, and specific voting protocols, based on blockchain transactions. These protocols will include zero knowledge proofs, Crypto Conditions [17], and Interledger Protocol [25] technologies to provide proper incentive for all participants to engage in the automatic creation and use of on-demand, isolated or public blockchains, ones that rest on a secure, public foundation.

The autochain protocol will also provide the ability to place a value on the data generated by the autochain's operation. This value will then either be released to the public or paid out, through rewards, to the participants in the creation of that data. This creates a model that incentivizes release of learning and information to the public – also making applications which do so much less expensive to run.

For Verus poll applications, validation will default to a form of proof-of-stake (PoS), where all members of the chain are stakers. This is because that security model matches exactly the interests of the majority of voters in having an accurate record. Because PoS does not involve powerful mining equipment or competition to solve a cryptographic puzzle, it can operate on much less powerful hardware– even on mobile devices. Since voters, themselves, secure the chain simply by running the voting software, poll chains are generally not affected by activity on the main Verus chain (except when posting interledger transactions). In addition, an autochain's operation, independent of scale, adds no overhead or congestion to the main Verus chain's transaction load – besides the transactions to provision the poll chain and post results back to the main chain itself.

Since we want to be able to create autochains from clients that may not be running full nodes or even own stable IP addresses, full nodes will be incentivized to provide reliable static node support for autochains. Once they provide this support, they will be recognized

by validators on the autochain. They will then be able to use this recognition to boost their reputation for selection in future lotteries.

By advertising autochain node support on the main blockchain – along with a stable payment address via a transaction – the node IP can be selected in a random lottery. The Eigentrust ratings of all nodes can then be recorded by all other nodes that serve the chain at regular intervals. Nodes which consistently offer better QoS will get better ratings. Nodes that get poorly rated will still share in the node rewards for the current blockchain – but will be less likely to be selected in the next node lottery.

3.2 The Virtue Autochain

3.2.1 Roots of Trust in Distributed Consensus

The Bitcoin [14] proof of work implementation introduced the world to systems that establish trust, not due to belief in any individual human's behavior, but based on the tendency of humans to act in their own self-interest.

The Bitcoin protocol rewards winners of a contest, referred to as "miners", in statistical proportion to the computing power they employ to "mine" for Bitcoin, as compared to the total computing power of all miners. For every block of validated transactions added to the Bitcoin blockchain, someone wins a competition to earn the right to define that block while adhering to specific, verifiable rules. The competition involves verifiably solving a statistical mathematical puzzle that is specific to exactly that block, meaning the same work cannot be reused on any other block. The winner of each competition earns the right to accurately process the next block of pending Bitcoin transactions and to claim a specific measure of Bitcoin, called the "block reward" (plus fees of all transactions in the block that was processed).

Mining competition serves to determine the amount of computing power a would-be attacker would have to control in order to mine of blocks that were not earned according to the intended rules, execute transactions that might otherwise be rejected, and prevent certain transactions from executing altogether. Based on the way that miners achieve consensus on what is the correct chain, forging a false Bitcoin chain to achieve this would require an attacker to control more than 50% of the total power of all miners – both honest and byzantine.

Since, right now, it is likely infeasible for any single actor to mount such an attack against Bitcoin, the largest network of hash power today, this type of competition, called proof-of-work (PoW), currently manages hundreds of billions of dollars of value, sends transactions of that value to anyone, anywhere, at any time, and does this all without any company, bank, or trusted third party, of any kind.

In addition to enabling secure management of large sums of money, PoW has also sparked a mining arms race and significant investment in computing power to claim Bitcoins and other cryptocurrencies, creating the equivalent of the world's largest distributed supercomputer doing nothing but the same calculations, albeit on different data, over and over again. As a result, one side effect of the public blockchains' PoW security system is massive power consumption and significant ecological cost.

Due to this, a great deal of research and development has gone into alternative consensus mechanisms [11] [12] [13]. Most serious PoW alternatives center around the concept of proof-of-stake (PoS) – the idea, that by putting something at stake commensurate with the value being transacted in the transaction flow, a rational staking person, known as a "validator," will choose good behavior within the ecosystem, so as not to lose their "stake."

Even though large systems are being built that depend on 100% PoS, there remains controversy over its robustness when users have incompatible interests, or when an attacker's stake is not valued as highly as the perceived value to be gained by corrupting the system.

Most modern PoS and PoW systems make a **fundamental assumption** about all miners; that they are either 'byzantine', and intend to compromise the system as a whole, or

'rational', in statistically the same proportions for all participants equally. One need only consider that if there was a way to learn a more accurate statistical function for each participant's probability of being **either** byzantine or rational, using such a function to determine who participates in the system's validation would improve its resistance to attack.

In fact, any accurate method of recognizing those who were attempting to strengthen the system – and thus who could statistically be more trusted than a byzantine or even average participant – could both decrease the power consumption and ecological cost of blockchain security and further strengthen resistance to attack. This then brings up the question: **"How do we recognize trustworthy participants?"**

Some methods studied and proven effective are the Eigentrust reputation management algorithm [20], and its improved derivative the Eigentrust++ mechanism implemented in the NEM blockchain [22]. Originally designed by Sep Kamvar, Mario Glosser and Hector Garcia-Molina in 2003, the Eigentrust algorithm is built to function on peer-to-peer networks. It aims to isolate byzantine actors in said networks by assigning each peer a public, global trust value based on their history of activity – clearly displaying a form of "rating" for each peer. Those with lower ratings are shown not to be trustworthy, and thus, are interacted with less by their peers.

Simply put, the algorithm assumes that if any given peer a trusts any given peer b, then it would also trust the peers trusted by b. Every peer calculates a local trust value for each other peer it has interacted with, based on the either satisfactory or unsatisfactory transactions it has had with each one. These local trust values are determined by each peer, and when peer a wants to know if they should trust peer b, say, before making a transaction, peer a would ask all other peers it knows to report on their local trust values of peer b and weigh their responses according to the trust values peer a has for each of them.

3.2.2 Proof of Virtue (PoV) Reward System

Using a model similar to the EigenTrust algorithm, Verus will introduce the idea of Proof of Virtue (PoV) **enhancement** to the PoS algorithm initially released in **Phase I**. In PoV, we intend to weight staking contests with both Verus stake and "Virtue", a special "currency", expected to be tracked on a Virtue autochain, and used as a **trust rating** of identities within the network.

In a similar way to the "amount of stake" used when staking, Virtue will add another component to the probability of being selected to process a block of transactions on the Verus Network. Its probabilities upon introduction will not change returns on the Verus PoS system, but addresses will gradually be able to improve earnings with a Virtue weighting – **based on recorded activity and behavior.**

In order to ensure that Virtue is both a rare and valued property, which drives correct behavior, it will neither be purchased, nor sold. Verus holders will have opportunities to earn a small measure of Virtue, which can then be further increased by staking the Virtue itself, when attesting to a fact for the network, validating information, or providing another measurable benefit to the network.

Since Virtue will have intrinsic value on the Verus network, there will be methods for transferring it through wallet ownership – for cases of probate or other necessity – but such behavior as a tool for trading in virtue will not provide a secure or intended method of

exchange.

The Proof of Virtue model will effectively be a **modified** form of PoS that is not based just on monetary value, but also on long term measurement of contribution, which will affect trustworthiness and earning power in the Verus network. The intent is to provide opportunities for more people to participate in the growth of a **positive functioning network** and to prevent potential attackers from being able to easily buy their way into an attack, adding yet another layer of security.

We expect to keep the PoW component as part of Verus security system for an indefinite period of time. We also expect to leverage the dPoW security of Komodo and its notarization into the Bitcoin blockchain as well.

3.3 Polls, Voting, and Elections

3.3.1 The Importance of Secure Polling

A great deal of research has delved into the best way to achieve confidential, verifiable, and transparent (CVT) elections electronically – with a few notable systems actually used in real life situations. These attempts, however, have been plagued with significant limitations, limitations that almost always risked either revealing the identity of a voter and thus eliminating confidentiality, or allowing attackers to impersonate legitimate voters, and put the validity of the vote into question [1].

This was shown in one of the first majorly adopted North American electronic voting systems, the Diebold AccuVote TS, which was announced in the year 2000 [4], at a time when, as a result of the Florida 2000 presidential election, the general public began to recognize numerous flaws in the widely-used punch card voting system [2]. Despite multiple studies discussing electronic voting systems – studies which clearly warned of security risks such as insider threats, issues with auditing, and network vulnerabilities – by 2004, the Accuvote TS system was deployed for major political elections in 37 US states [2], resulting in multiple serious problems that impact the legitimacy of numerous election results since, due to inherent flaws in its design that were never fully addressed [3].

Firstly, the "solution" that the system introduced to deal with voters submitting multiple ballots – and to solve the ballot anonymity problem – was to issue a single personal voting 'smartcard' to each vote. This smartcard, unfortunately, did not contain a complete identity, but instead contained a common election key among voters that voting machines simply checked to determine if the card belonged in the correct election [2]. The lack of any cryptographic unique identification on these cards was a significant security flaw, as user-programmable smart cards and readers almost instantly became commercially available on the internet for reasonable prices – making it extremely easy to mass produce homemade copies. Furthermore, due to the lack of a truly secure boot loader, operating system, or application, the system had numerous potential attack vectors, many of which were quite simple for an adversary. For example, any attacker with access to the operating system had the ability to modify ballot program files through the standard Windows Explorer application already included on each machine.

These serious failures in previously adopted electronic voting systems have literally **changed the course of history**, affecting the outcomes of major political elections that

decide the state of world politics. This highlights the importance of creating a system that is truly confidential, verifiable and transparent, and indicates the magnitude of one problem we plan to solve with Verus.

3.3.2 Running, Recording, and Scaling Polls - Transient, PoS Chains

Using autochains, Verus will dynamically create parallel voting chains on demand for each poll in its ecosystem. This creates improved, isolated security and allows for scaling in a virtually unlimited manner.

The Verus wallet, which will be used for voting, will also have the ability to validate blocks through proof-of-stake, blocks which, when enabled, will allow voters who are validating the election to earn rewards. These rewards will either be paid in Virtue for intrinsic polls, when available, or from outputs of the payment transaction that instantiated the poll. When a poll is complete, results are posted back to the main Verus chain. The chain used to run it can then be archived or deleted, and it is typically abandoned.

3.3.3 Poll Content and Distributed Hash Tables

While blockchains make excellent databases for permanent, non-repudiable records and public key management, they are not the best solution for storing large amounts of data, due to their massive duplication across the distributed network. Instead, a better model for distributed storage would look something like the Interplanetary File System, or IPFS, which is a peer-to-peer file system based on the Kademlia [18]19 protocol in an implementation that currently works well enough to use as storage for create/read/delete operations using hash commitments on the Verus blockchain to represent and index content.

While we are evaluating the IPFS, we are also looking at other distributed hash table solutions, such as Open DHT [26]. A key requirement for the Verus network is that mobile devices be able to participate in its operation. Open DHT is an efficient C++ library intended for use with small devices. IPFS does not yet seem to have much support for mobile at this time, though it does have a Javascript library that can be used from the browser or mobile apcancations [24].

Until IPFS is supplanted with another implementation, Verus applications will assume that when off-chain content is needed, it should be retrieved from IPFS, and for future compatibility, we will add a storage and version specifier. In order to store supporting poll or other content for dApps, the content owner must ensure that the content is pinned in IPFS, until it is no longer needed. By **Phase IV**, or earlier if there is significant demand, we expect to either support or provide a blockchain-metered storage solution that is more integrated and automatic to use for poll makers. Two options for such a system include a cross-blockchain integration with a decentralized payment/pinning solution, or to offer a simple service that charges in Verus to store and automatically pin content for specific lengths of time in IPFS.

3.3.4 Voting Models and Types of Polls

Verus will support numerous types of polls, those that use weighted or unweighted voting, that seek to expose truth or opinions, and polls that rate and choose collectively. In order to

create a sensible taxonomy for polls, one that people will understand, we define the following types of voting:

1. Multiple choice
2. Weighted multiple choice
3. Ranking
4. Rating

We also define the following types of polls:

1. Polls to classify
2. Polls to select
3. Polls to rank
4. Polls to rate

We do not expect Verus to support all voting and poll types in the early phases. In fact, until **Phase IV**, the Verus core team plans to focus on known requirements of multiple choice and weighted multiple choice voting. During or after completion of **Phase IV**, expanding support for voting and poll types will be higher priority.

Each type of Verus poll may support confidential or public voting, live or delayed poll results, online or offline vote authorization, and lottery selected vote authorization. We will roll out working vote models in phases, as well, initially providing solutions for weighted and unweighted selection voting, which fits well for both political and opinion polls as well as content classification and ID verification.

3.3.5 Intrinsic Polls

Once Virtue is activated, the Verus network itself will run specific, intrinsic polls that will be regularly available to identities with Virtue, based on a statistical lottery and weighted by their actual measure of Virtue. Participating in these polls, which will be used to classify content and perform basic, human verification functions, enables participants to help the system classify and rate content across Verus applications, enabling a rating system rather than censorship to allow each voter to set preferences for the ratings of polls they may wish to see.

Miners who provision autochains for these intrinsic polls will get the same rewards as miners setting up autochains for commissioned polls. The content for these polls will be taken from other polls and will be used to classify and rate those polls for easier discoverability and the user experience of poll services.

3.3.6 Privacy

There is a fair amount of confusion among the public about privacy on the blockchain. One common belief is that Bitcoin transactions are private, when in fact all Bitcoin transactions, as well as the public addresses between which value is transferred, are public information [21]. As with the Verus token and its blockchain, poll autochains will also support zk-SNARKs to ensure that votes which have not been disclosed by a voter remain provably anonymous to all other parties, while still being verifiable by the person who holds the keys to the address that cast them. Currently, based on standard smartphone hardware specifications and the memory requirements of zk-SNARK transactions, mobile phones cannot generate z-transactions. In coming releases of new Zcash technology this year, the z-transactions will be more efficient and still provide the same privacy guarantees. When these technologies are available, we will work to support them on the Verus chain, as well as use the increased efficiency to enable z-transactions and all Verus features on mobile devices as soon as possible. Until that is available, we will work to provide a mobile wallet for Verus transparent transactions.

3.3.7 Content Classification

While Verus is designed to provide invaluable tools to people across the world, any system without censorship will also inevitably allow the underbelly of humanity to show through. That means that while we would want all people to participate, we recognize that in order for that to provide the most positive experience for all participants, we must use the Verus poll system itself to recognize, rate, and classify the content on its network, enabling those who provide such a service through participation in polls to earn, while enabling the selective filtering of content based on value judgements and classification of honest populations.

While we recognize that the best early use case for such capability is in rating and classifying non-intrinsic polls themselves, the ability to classify content is a fundamental strength that Verus will increasingly develop over time. At first, we expect such classification to be applied to relatively simple tasks for humans, such as identifying toxic comments, hate speech, or categories of information. Even though these types of classification systems have largely employed machine learning systems running against privately curated training data, our vision is for Verus content classification to enable people to earn, as they classify content better than any machine learning system, but in a way that can be followed by the best machine learning algorithms, and generalized at scale.

3.3.8 Truth vs. Opinion

When classifying content, especially when you start to consider challenging classifications, such as misleading, propaganda, generally accepted fact, accurate versions of history, vs. classification of what should generally be accepted as toxic or rated content that people could reasonably answer objectively, one must consider the difference between fact and opinion. Selecting the winner of an election is a matter of expressing an opinion. Classifying content according to level of toxicity is a question of determining the factual answer to the question: what does the majority of the voting population believe the classification to be. Determining

whether an image in one photo is the same image as that from another is a question with an objective, if not determinable answer.

To ensure that Verus polls support polls that attempt to discover accepted truths, facts, and even credible emerging arguments, Verus voters will often be able to stake and either earn or lose Verus and in later phases, Virtue, by answering in a manner that matches consensus. For other polls, which are recognized as opinion polls, all answers are equally valid contributions.

3.4 Random Sampling as a Service

The Verus system will randomized selection for many of its functions, similar to its **Phase I** PoS block validation system. Blockchain lotteries will be generalized and used for selection of participants in randomized polls and in other cases where pseudo-random sampling is desired. The general principles behind Verus lotteries are the same, regardless of whether something similar to Algorand [29] for guaranteed selection or an original Verus algorithm based on difficulty, such as Verus PoS is used for selection. Lotteries use the blockchain as a random oracle and are based on the assumption that the exact value of a specific block hash from a past block in the chain cannot be modified by a byzantine participant to weigh the odds in their favor.

3.4.1 Participating in a Lottery

Since the Verus network is completely decentralized, all lotteries, including PoS, are potential, and there is no central server selecting and sending messages to those who qualified in a lottery. When lotteries are implemented as a general feature, if you would like to participate in any Verus lottery or poll that is distributing tickets via lottery, you must first determine that you qualify for the poll's requirements. You will then need to submit a transaction that proves that you have a valid claim and spend the output transaction of the lottery ticket to your address using that proof. By default, that spend will authorize you to be a validator on the autochain for the poll you are participating in, by providing you with the currency of that poll. You may validate the poll by leaving your wallet running, getting potential rewards in Verus or Virtue for both activities.

Lotteries will be useful for selecting subpopulations based on identity and claims people make about themselves or that others claim about them, which they are willing to share. While detailed discussion of Verus identity is beyond the scope of this paper's release, the Verus system will provide self-sovereign identity with the ability for identities to make very flexible claims and have them attested to by other identities. For example, a lottery may look for males between the ages of 18 to 25 among identities willing to share that information, which will even be possible to verify as strongly as through a validated passport or driver's license. Other polls may not require the same evidence backing claims. Polls may also have restricted voting, such as local political polls, where verification and authorization for the poll must be provided explicitly, often by mail to a physical address or in person in the form of a QR code or electronically delivered transaction to your Verus wallet.

3.4.2 Proof of Stake Lottery

A block validation lottery is a somewhat different form of lottery which allows you to claim the right to validate a block and its associated reward based on proof-of-stake (for autochains) or 50/50 PoW/PoS (for the main chain). The lottery requires that a prior block hash 100 blocks past combined with a qualifying UTXO of the destination public key hashed with the staking block height and then divided by the UTXO value must be under the current proof of stake validation difficulty. By dividing by the UTXO value, Verus weights proof-of-stake or proof-of-virtue in proportion to the amount of stake or virtue in the UTXO. If no one who qualifies to validate a block is online or no one responds with a submitted block validation in a certain amount of time, the blockchain will wait until someone mines a block using proof-of-work.

3.4.3 Ticket Harvesting Lotteries

There will be multiple ways to acquire tickets to participate in any particular poll. The manual ways emphasized so far may include receiving a poll with QR code in post or email or perhaps directly from someone taking a poll. In addition to these direct ways to receive a voting ticket, Verus will enable polls to be posted on the poll blockchain as transactions, the tickets for which are distributed from its outputs by lottery. This will work when the Verus ID functionality is available, and will enable polls to require presence of specific claims or attestations on an ID, as well as the number of participants to select in the poll by creating one transaction output for each, which is spendable by a cryptocondition [17] that defines the lottery conditions and random difficulty. The difficulty determines how often a particular block will match any attempt to harvest a ticket from that block.

3.5 Machine Learning Integration

Verus enables humans to share and expose human knowledge through voting mechanisms to make decisions, express opinions, and classify accurately. The data used for that classification, as well as the classifications or decisions themselves provide an ideal source of training data for machine learning systems across any domain of human knowledge or opinion. As corporations gather, sequester, and learn from massive amounts of personal data in order to compete and boost their bottom line, rather than accrue to society's benefit, these massive private databases serve to affect and influence human populations by knowing more about them than others, or even than they know about themselves.

Since Verus enables humans to use voting mechanisms to classify and make decisions, and stores the results of this process in IPFS and on the blockchain, we will train machine learning algorithms on these results over time, allowing them to eventually classify and make decisions consistent with human decisions and values at an even higher scale. This will enable Verus to perform certain tasks automatically, such as the recognition of false or incorrect identification data among participants in the system, the initial classification or rating of certain content that can be overruled, but may not be appropriate for all audiences, and much more. The end result of this is a more secure and trustworthy network, built on consensus and trustless principles, leveraging worldwide human knowledge as a base of information and machines for scale.

A major Verus advantage over almost any other system in its use of Machine learning on human data is the innate privacy it provides to those whose data is used for learning, along with the default that all resulting information be made public, unless those making polls pay the Verus community a going rate to keep the data for themselves. In today's society, the goal of most machine learning systems is to match consumers with products they are likely to buy and/or manipulate them into an actual purchase, thus generating revenue. This creates a situation that can create unintended negative consequences when algorithms disregard any positive or negative effects its predictive abilities may have on society [23]. For example, if the algorithm recognizes that people with depressive episodes are more likely to gamble, and thus buy airline tickets to gambling-oriented locations, such as Las Vegas, it will advertise those airline tickets to those people. In the Verus system, machine learning algorithms will be able to learn from human beings, while being limited to accessing only the data users permit, and at the same time, being unable to easily target any specific individual.

3.6 Verus Mobile Polls

In addition to a desktop wallet that supports Verus and other Komodo platforms or compatible applications, mobile support is a high priority and will provide yet another layer of convenience for users, enabling them to use, earn, and spend Verus in everyday life. We intend to provide a Verus mobile experience that functions as a wallet, self-sovereign identity, provides easy access to the polling and earning applications, and is intended to server as an extensible application browser, capable of supporting additional applications that leverage the Komodo or compatible ecosystems. While we have already begun thinking about and storyboarding design and development of user experiences, we do not expect any mobile experience to be ready until completion of **Phase II** at the earliest. Even when the first mobile experiences are ready, they will not include the support for private transactions that we intend to enable as a foundation for confidential, mobile polling and communications. To get a feel for the way we envision Verus experiences working in practice, below is an example of some poll screens from early designs.

4 Implementation and Roadmap

After researching options from building the technology ourselves to leveraging unreleased advanced projects, to using one of the few well thought through blockchain application platforms, we decided to start building the Verus project as a friendly fork of Komodo and its asset chain technology, enabling us to both become a contributing member of and also enhance and extend the Komodo platform as Verus builds blockchain platform technologies and real world applications of PBaaS.

By leveraging proven zk-SNARK zero knowledge privacy technologies and Komodo's delayed proof of work (dPoW), which notarizes the main Verus blockchain into Komodo's blockchain, which is then notarized into the Bitcoin blockchain, we underpin Verus with a foundation of state-of-the-art privacy, security, and interledger transaction capabilities as our baseline. Above that, we have already developed advanced features that further enhance security, improve decentralization, and prepare for the implementation of autochains and proof of virtue.

We will maintain our own open source fork of the Komodo platform and collaborate with the Komodo team when possible to develop new features or capabilities that can be accessed cross-chain by projects across the Komodo platform and even other blockchains that are simply compatible with Komodo's Interledger Protocol implementations. We will maintain our development as an open source project and contribute back to the broader open source community, as we leverage the contributions of others that have made it possible for us to begin realizing our vision as a community.

All that said, roadmaps are never perfect. Unless we significantly underpromise and prioritize a date so highly that no feature or capability is important in and of itself, targeting timeframes and milestones without specific target dates allows us to be ambitious in our goals and adapt to a changing world as we deliver. Even in **Phase I**, which we have delivered as of the writing of this paper, we adjusted our plan, and in an uncharacteristic turn of events, delivered even more than we had originally planned. We do not expect that to always be the case, and we will focus more on correct and complete phases than specific dates when possible. Sometimes, as with any major endeavor, the right choice is to recognize an

unexpected opportunity or obstacle, take two steps to the right or left, and only then proceed forward.

For the purpose of this whitepaper, we will discuss four phases of the Verus vision. We do not see these four phases as a completion of the vision as much as four phases of an ongoing mission to advance Verus and its contribution to society that we can currently express. From its inception and throughout the project, the Verus chain will serve as our fungible value chain, and use of Verus tokens on this chain will be the core value around which we continue to develop the Verus vision.

The first Verus chain, which is already available at the time of this white paper's release, includes zk-SNARKs for privacy, both transparent and private addresses, a brand new CPU-mineable hash algorithm for proof of work, a brand new proof of stake algorithm, and a unique emission schedule among fair launch cryptocurrency projects. Each phase of our project's development will introduce capabilities and experiences that provide independent value on their own, further leverage Verus Coin to power operation, and create a foundation upon which to build the next phase.

We intend and hope for the Verus project to become a worldwide, inclusive community effort, which welcomes and rewards those who contribute. Although we articulate these first four phases here, we see these phases as just the beginning, and we hope you will join us by participating in and contributing to the Verus project to make our world a better place.

4.1 Phase I – Mainnet – 50/50 PoW/PoS – Time Locks – Fair Launch

In phase one, the Verus main network began with a slow start and 15 minutes advance notice at 7:15am GMT, Monday, May 21st, 2018.

The Verus network began emitting first 0 block reward, rising linearly each block over the course of 7 days until block 10080 to its peak of 384 coins per block. The Verus emission schedule is as follows:

Era 1:

Block 0 - 10080 reward: 0 to 384, rising linearly and changing each block

Era 2:

Block 10080 - 53279 reward: 384
Block 53280 - 96479 reward: 192
Block 96480 - 139679 reward: 96
Block 139680 - 182879 reward: 48
Block 182880 - 226079 reward: 24

Era 3:

Block 226080 - 1277279 reward: 24
Block 1277280 - 2328479 reward: 12
Block 2328480 - 3379679 reward: 6
...halving indefinitely every 1051200 blocks (approximately 2 years)

We also added another fair twist on the launch that we believe will contribute to a more stable value growth in the Verus currency, without any unusual risk of dumping by any single party. During the first 5 months, Verus will have an accelerated reward curve for mining and staking, with a halving every month. During the first two months, when the block rewards are at or above 192 Verus, mined blocks will have time locked coinbase transactions, preventing spending, staking, or transferring of those coins for a period that varies from approximately 3 months after the genesis block, based on numbers of blocks, to 2 years and 3 months. These time locked coins provide a smooth release of the Verus currency supply as well as incentive to all of their owners to support the long term success of the Verus network and project.

From the very beginning of its operation, Verus operated with a dual proof of stake / proof of work mechanism for all participants. Verus mining with the VerusHash algorithm, as of this point, is a CPU-only algorithm, designed with a Haraka512 V2 [30] core to be quantum secure and to maximize performance on modern CPUs. While VerusHash was designed to be CPU-optimized and did not begin life with GPU or ASIC miners capable of beating CPUs in its mining, the Verus developers have no illusions that it is possible someone might develop either a GPU or ASIC-based miner that could be meaningfully superior to CPUs.

To ensure that such a solution if developed, remains open, jl777, lead developer of the Komodo Platform, has offered a 1 Bitcoin bounty to any developer who can make and publish in an open manner, a GPU miner for the VerusHash algorithm that can significantly outperform the current CPU miner. In order to win the bounty, the source code for the miner must be available under open source licensing. To be eligible for the bounty, any implementation must be able to perform 5x better than a modern, high thread-count, high clock-rate CPU on a GPU costing under $1000. By bringing up the Verus network this way, we established an immediate, baseline set of functionality, above which we can build out our project and community around a functional coin and project, as we stay in sync with advancements in the underlying Komodo platform.

Verus phase one release was pre-announced on Bitcointalk [28] with zero premine, and team members mined and staked along with everyone else to generate coins. In addition to coins mined by individual team members for their own benefit, the Verus developers intend to donate most of their mined and staked earnings to a community Verus foundation along with other donating community members who will join us, in order to support the ongoing growth and project development by core developers and the community for years to come.

4.2 Phase Two – Autochains, Polls, Verus Mobile

As soon as we completed phase one, we entered phase two, which includes release of this white paper, activation of the community through donations of some of our mined coins, and a push to finish planning for, development and release of mobile support, autochain capability, and the ability to setup, and then easily run CVT polls at scale.

4.3 Phase Three – Identity, and Proof of Virtue

In phase three, we plan to further improve the mobile experiences, implement the identity system, including support for strong, decentralized identity and attestations, separating photo IDs and other photo content into components that can be separately verified in unbiased human polls. At this point, we intend to support optional KYC strong identities, via both notarization of identities as well as poll-based identity verification. This will also be the first phase that supports Verus chain validation PoV enhanced PoS.

4.4 Phase Four – Integrated Machine Learning, Content, Storage

In phase four, Verus will begin to truly leverage the foundation built in phases one, two, and three by focusing on broad content classification capabilities for off-chain content, improving storage management support in its distributed hash table implementations, supporting storage monetization in some way, and providing open source, public implementations of machine learning systems that can learn from data on the Verus network to solve real world challenges of today and tomorrow.

5 Forward Looking Statements

This paper includes predictions, statements of intent, discussion of plans, estimates or other information that might be considered forward-looking. While these forward-looking statements represent our judgment and expectation of what the future holds, this is not an offer or solicitation to purchase any product, good, service, or security. All statements herein are subject to risks and uncertainties that could cause actual results of the Verus Project's development to differ materially. Furthermore, we intend to use the Verus blockchain as our open source development platform – contributing these technologies under permissive licensing for the betterment of society, not focusing solely on profit of anyone affiliated with the Verus project. You are cautioned not to place undue reliance, especially in any financial decision, on these forward-looking statements, which are subject to modification, update, or change for legitimate reasons both within or beyond our control. By expressing our vision and goals, the Verus Core developers are not obligating ourselves to revise or publicly release the results of any revision to these forward-looking statements in light of new information or future events.

References

[1] S. Estehghari and Y. Desmedt, Exploiting the client vulnerabilities in Inter-net e-voting systems: Hacking Helios 2.0 as an example, Proc. 2010 Electron. Voting , no. Section 4, pp. 027, 2010.

[2] Kohno, Tadayoshi, et al. Analysis of an Electronic Voting System. IEEE Computer Society Press, 2004, Analysis of an Electronic Voting System, http://avirubin.com/vote.pdf

[3] Hursti, Harri. "SECURITY ALERT: May 11, 2006 Critical Security Issues with Diebold TSx ." Black Box Voting, Black Box Voting, Inc, 11 May 2006, http://www.blackboxvoting.org/BBVtsxstudy.pdf

[4] Diebold Election Systems. AVTSCE source tree, 2003. http://users.actrix.co.nz/dolly/Vol2/cvs.tar

[5] Willet, JR, et al. "OmniLayer/Spec." GitHub, JR Willet, 24 Jan. 2017, http://github.com/OmniLayer/spec

[6] Stone, Andrew. "Bitcoin Cash Scripting Applications: Representative Tokens (OP_GROUP)." Medium, Medium, 16 Oct. 2017, http://medium.com/@g.andrew.stone/bitcoin-scripting-applications-representative-tokens-ece42de81285

[7] Abed, Gabriel, et al. "Colu Local Network." Colu Network, Colu Technologies DLT Limited, Jan. 2018, http://cln.network/pdf/cln_whitepaper.pdf

[8] Assia, Yoni, et al. "Colored Coins White Paper - Digital Assets." Google Docs, ColoredCoins, June 2017 , http://docs.google.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/edit#heading=h.wxrvzqj8997r

[9] "Ethereum Blockchain Congestion Triggers Outrage." Global Coin Report, Global Coin Report, 12 Dec. 2017, http://globalcoinreport.com/ethereum-blockchain-congestion-triggers-outrage/

[10] Sedgwick, Kai. "The Ethereum Blockchain Is Congested by Cats." Bitcoin News, Bitcoin News, 4 Dec. 2017, http://news.bitcoin.com/ethereum-blockchain-congested-cats/

[11] "Delegated Proof-of-Stake Consensus." Delegated Proof-of-Stake Consensus - BitShares, Bitshare, 8 June 2015, http://bitshares.org/technology/delegated-proof-of-stake-consensus/

[12] Buterin, Vitalik. "A Proof of Stake Design Philosophy – Vitalik Buterin – Medium." Medium, Medium, 30 Dec. 2016, http://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51

[13] Poelstra, Andrew. "On Stake and Consensus." WP Software, 22 Mar. 2015, http://download.wpsoftware.net/bitcoin/pos.pdf

[14] Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." Bitcoin.org, http://bitcoin.org/bitcoin.pdf

[15] Lee, James. "Komodo: An Advanced Blockchain Technology, Focused on Freedom." Komodo Platform, Komodo, 12 Mar. 2018, http://komodoplatform.com/wp-content/uploads/2018/03/2018-03-12-Komodo-White-Paper-Full.pdf

[16] Mercer, David, and Duke Leto. "HushList Protocol Specification." Git Hub, http://raw.githubusercontent.com/leto/hushlist/master/whitepaper/protocol.pdf

[17] Thomas, S., and R. Reginelli. "Crypto-Conditions." IETF Tools, 9 Jan. 2017, http://tools.ietf.org/html/draft-thomas-crypto-conditions-02

[18] Maymounkov, Petar, and David Mazieres. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." Parallel & Distributed Operating Systems Group, http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[19] Benet, Juan. "IPFS - Content Addressed, Versioned, P2P File System." IPFS, http://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

[20] Kamvar, Sepandar D., et al. "The EigenTrust Algorithm for Reputation Management in P2P Networks." The Stanford Natural Language Processing Group, http://nlp.stanford.edu/pubs/eigentrust.pdf

[21] "Some Things You Need to Know." Some Things You Need to Know - Bitcoin, 2015, http://bitcoin.org/en/you-need-to-know

[22] "NEM Technical Reference." NEM, http://nem.io/NEM_techRef.pdf

[23] Tufekci, Zeynep. "We're Building a Dystopia Just to Make People Click on Ads." TED: Ideas Worth Spreading, Sept. 2017, http://www.ted.com/talks/zeynep_tufekci_we_re_building_a_dystopia_just_to_make_people_click_on_ads

[24] "IPFS js-ipfs Javascript Github Repository" https://github.com/ipfs/js-ipfs

[25] Thomas S., et al. "A Protocol for Interledger Payments",http://interledger.org/interledger.pdf

[26] Savoir-faire Linux Inc. "A C++ 11 Distributed Hash Table Implementation," http://github.com/savoirfairelinux/opendht

[27] Goldstein, Jacob, and David Kestenbaum. "The Island of Stone Money." NPR, NPR, 10 Dec. 2010, http://www.npr.org/sections/money/2011/02/15/131934618/the-island-of-stone-money.

[28] Toutonghi, Michael. "[ANN] Verus (VRSC) - Zk-SNARK Privacy, CPU-Mining, 50/50 POW/POS, Fair Launch." Bitcoin Forum, Simple Machines Forum, 21 May 2018, http://bitcointalk.org/index.php?topic=4070404.0

[29] Gilad, Yossi, et al. “Algorand: Scaling Byzantine Agreements for Cryptocurrencies.” MIT CSAIL, 24 Sept. 2017.

[30] Klbl, Stefan, et al. “Haraka v2 – Efficient Short-Input Hashing for Post-Quantum Applications.” International Association for Cryptologic Research, 24 Dec. 2016.