File Verification Operations

Intro

This operations document is meant to guide the reader through the process of creating a tezos wallet, uploading a smart contract, & calling that smart contract using java. The smart contract stores names of files along with their SHA256 hash. The java code parses through files and uploads their associated hash to the Tezos blockchain.

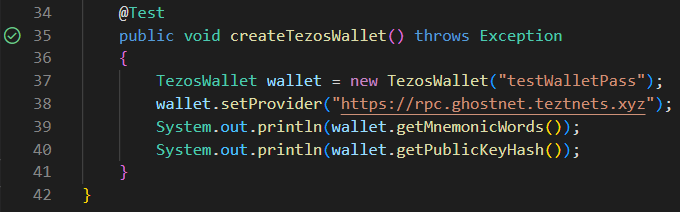
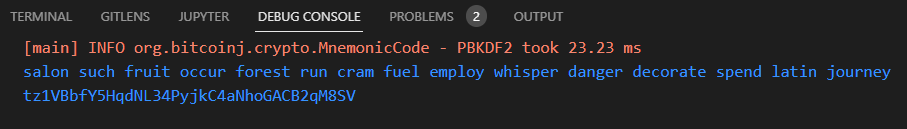
Prerequisites

* Visual Studio Code
  + Extension Pack for Java Extension
* Java 8
  + <https://www.guru99.com/install-java.html>
  + Check Installation w/ ‘java -version’ in cmd or VSCode terminal
* Note: This guide is made on a Windows machine

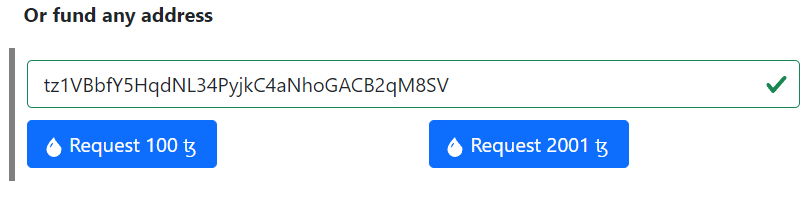
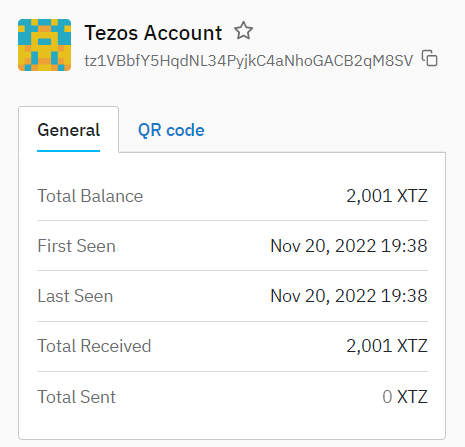
Download Github Repository

* Github Repository: <https://github.com/Corbo2000/TezosFileIntegrity>
* Open the Repository with Visual Studio Code
* FileIntegrityTest.Java has 3 Test Methods
  + testUploadFileHash
  + testVerifyFileHash
  + createTezosWallet

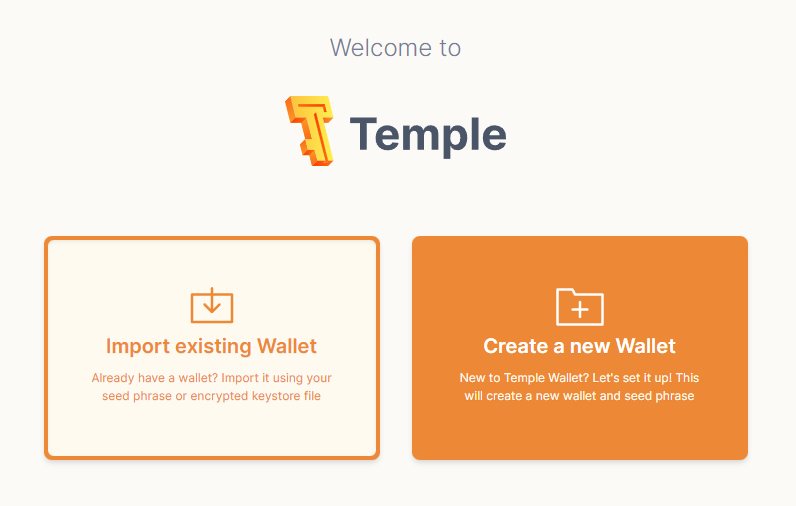
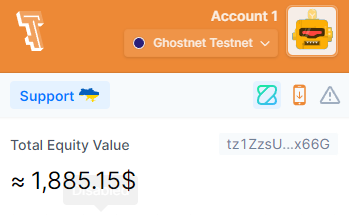
Create Tezos Wallet

* Create Tezos Wallet on Ghostnet Testnet to Use SmartPy Smart Contract
  + You can do this using external wallet or can use the createTezosWallet() method in FileIntegrityTest.java
  + Using the method will print Mnemonic Words & Public Key Hash
  + Make a note of the debug console output for **Mnemonic Words** & **Public Key Hash** as well as the **password** used on line 37. Default is “testWalletPass”

Use Faucet to Fund Wallet

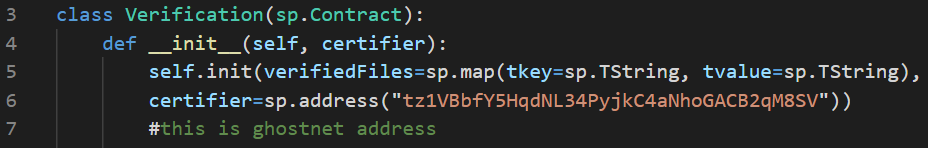
* Your Wallet Needs Funds to Operate on the Ghostnet Testnet
* Faucet: <https://faucet.ghostnet.teztnets.xyz/>
* Enter Public Key Hash and Request Funds
* Use <https://ghost.tzstats.com/> to Verify Receipt of Funds

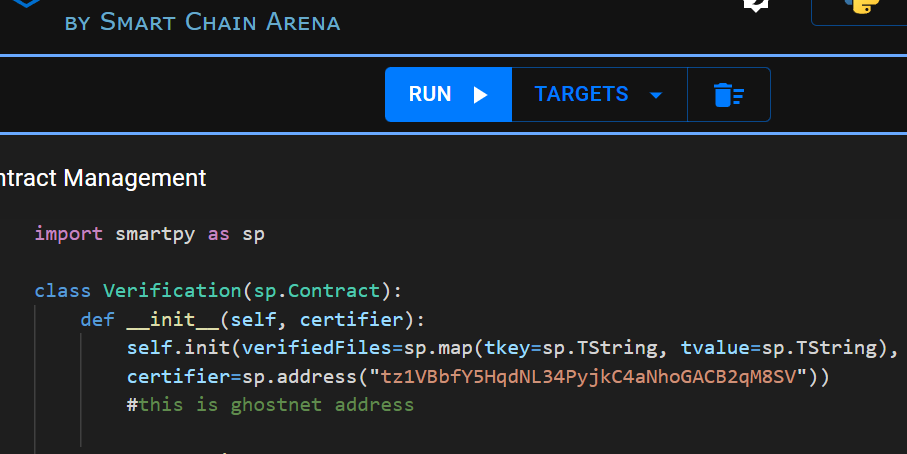
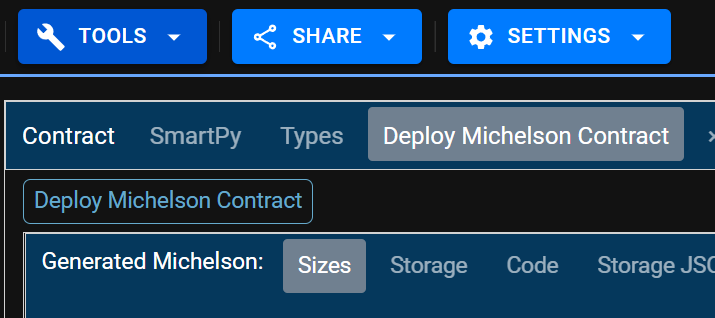
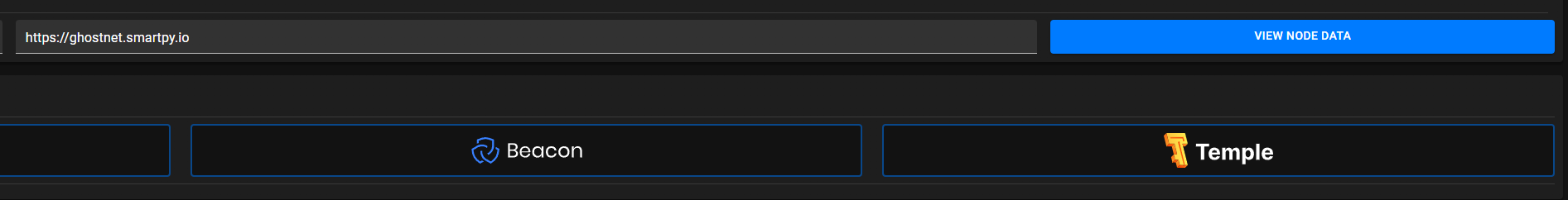
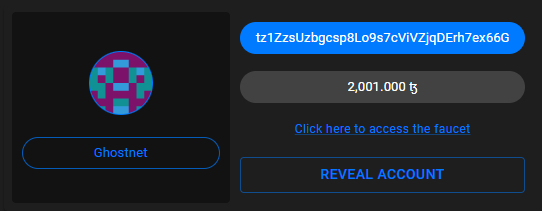
Sign Into Temple with Wallet

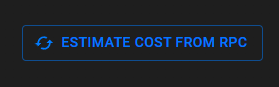
* Using the Temple Chrome Extension you can deploy smart contracts on SmartPy
* Add the Extension to Chrome and Create a new Wallet
* Save the Mnemonic Words and select a password. (This Wallet is solely for deploying the smart contract and is different from the wallet you created within the java test code.)
* Once again you will need to fund this wallet using the faucet: <https://faucet.ghostnet.teztnets.xyz/>
* The public key can be found by clicking here:
* Make sure you have selected the Ghostnet Testnet when you copy your public key
* Now that this Temple Wallet is setup we can proceed to deploy the smart contract

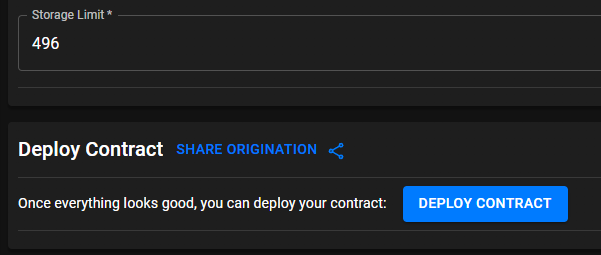
Deploy Tezos Smart Contract

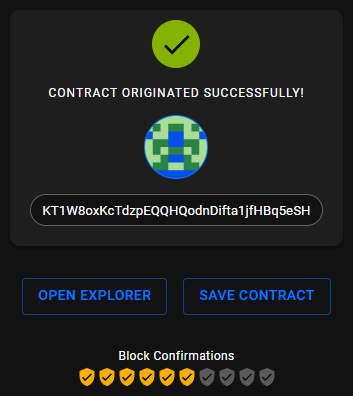
* Now we will deploy the File Verification smart contract from SmartPy.
* Smart Contract Link: <https://bit.ly/3Md4AHN>
* If you want to be able to call the certify entry point from your new wallet, you will need to change the certifier address within the Smart Contract.
* You can do so by replacing lines 6, 29, 37 & 38 with your Public Key Hash:



* Now we need to run this smart contract. 
* Next we need to Deploy Michelson Contract
* The Ghostnet testnet network must be selected
* Then the Temple wallet must be linked
* A popup will appear asking to connect your Temple wallet to SmartPy.io, select the connect button
* Now you must reveal your Temple wallet to smartpy. Select Reveal Account and sign the popup that appears. 
* Select Estimate Cost From RPC to determine gas fee



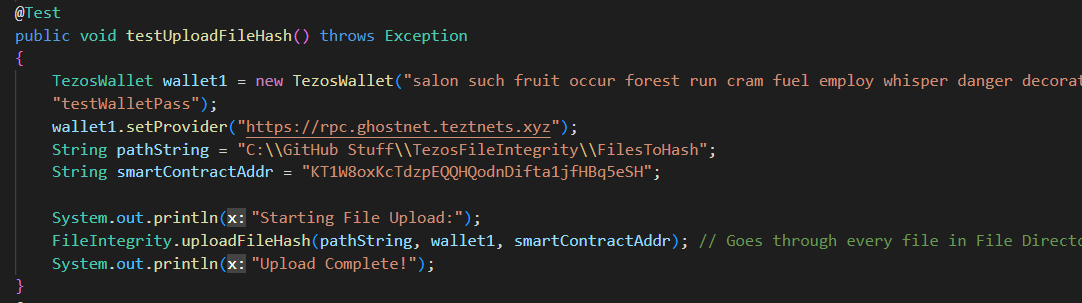
* Now we are ready to Deploy Contract
* Click Accept then Sign the Temple Wallet popup
* Given that the testnet is currently online, you should see block confirmations.

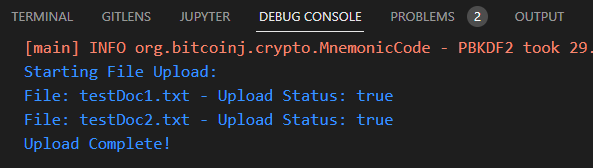
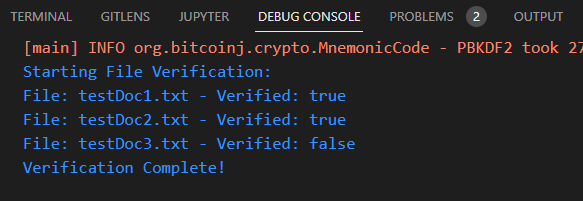


* Select Save Contract and make a note of the Contract Address. We will need this address within the Java code.

Call Smart Contract With Java

* Now we can test the File Integrity smart contract using the TezosFileIntegrity java project.
* Navigate to the “*{root directory}\*TezosFileIntegrity\src\test\java\fileintegrity\FileIntegrityTest.Java” file within Visual Studio Code to test the smart contract.
* Earlier we used the createTezosWallet method but now we need the other two
* First we will use the testUploadFileHash method to upload files to the blockchain
* A few fields will need to be updated to match your wallet / directory
  + Change the Tezos Wallet Field on line 17 to include your personal Mnemonic Words and Password.
  + Change the pathString field to match your local directory of FilesToHash. This path can lead to any other folder but keep in mind the time it takes to upload files. Starting with 1-2 files in the directory is a good place to start.
  + Insert your Smart Contract Address on line 21.



* After changing those fields run the test
* There will be an output in the debug console
* Now update the information in testVerifyFileHash. By running this we can see that the files were successfully uploaded. Change the contents of the files in FilesToHash (or whatever directory you are pointing to via pathString) to see that they will not be verified due to the change in their hash. In the case below I added a third file then ran the test. 
* Use <https://better-call.dev/> to see the results of these tests on your contract. Just search the smart contract address and look at the different entrypoint calls. You can select the Storage tab to see what files are currently stored on the blockchain.

References + Other Links

* <https://www.guru99.com/install-java.html>
* <https://faucet.ghostnet.teztnets.xyz/>
* <https://ghost.tzstats.com/>
* <https://smartpy.io/>
* <https://github.com/TezosRio/TezosJ_plainJava>
* <https://chrome.google.com/webstore/detail/temple-tezos-wallet/ookjlbkiijinhpmnjffcofjonbfbgaoc>

Code

* Github Link: <https://github.com/Corbo2000/TezosFileIntegrity>
* Smart Contract Link: <https://bit.ly/3Md4AHN>