03/20/21 ; 05/24/21
Pakistani province plans to build pilot crypto currency mining farms
Link Pakistani province plans build pilot crypto currency mining farm
By Umar Farooq ; https://www.reuters.com/journalists/umar-farooq
FUTURE OF MONEY
MARCH 18, 202112:01 AM UPDATED 3 DAYS AGO © www.routers.com 2021

PESHAWAR, Pakistan (Reuters) - The Pakistani province of Khyber Pakhtunkhwa is planning to build two hydroelectric-powered pilot “mining farms” to capitalise on a bullish global cryptocurrency market, a minister overseeing a new government crypto policy told Reuters on Wednesday.

The announcement comes as cryptocurrencies are gaining mainstream acceptance, with Bitcoin’s price reaching record levels as investors such as Elon Musk pour funds into it, and the first big U.S. bank, Morgan Stanley, offering its wealth management clients access to bitcoin funds.

Crypto mining farms involve large investments in computer data centers which require vast amounts of power.

Pakistan has formed a federal committee to formulate a new crypto policy, even as neighboring India is planning to ban cryptocurrencies entirely. The cost of the mining project has yet to be determined.

“People have already been approaching us for investment, and we want them to come to Khyber Pakhtunkhwa, earn some money and have the province earn from that as well,” Zia Ullah Bangash, advisor to the provincial government on science and technology said.

Both mining and trading in cryptocurrencies currently exists in a legal grey area in Pakistan, though, and federal authorities would have to provide a clear path towards legalizing the sector before it could be formally opened to investors.

In 2018, the State Bank of Pakistan said cryptocurrencies were not legal tender and the regulator had not authorized anyone to deal in them in the country. Pakistan also is currently on the global Financial Action Task Force grey list, and one of the areas the global money laundering watchdog has asked Islamabad to better regulate is cryptocurrencies.

Still, mining and trading in cryptocurrencies thrives in Pakistan, with apps like Binance and Coinbase among the most popular downloads in the country, according to web analytics company SimilarWeb.

“It’s really just our government that is not participating right now, people all over Pakistan are already working on this, either mining or trading in cryptocurrencies and they are earning an income from it,” Bangash said. “We are hoping to bring this to a government level so things can be controlled and online fraud or other scams can be prevented.”

Reporting by Umar Farooq; Editing by Kim Coghill

Our Standards: The Thomson Reuters Trust Principles.

The ability to code a ERC20[i] token to securitize[ii] a cash flow stream[iii]?

Use case definition...{#_ _ _ _}

I. Zia Ullah Khan Bangash[iv]
II. MoneyGram
III. P2P
IV. UnBanked
V. Khyber Pakhtunkhwa, MoneyGram local site
1.PAKISTAN CURRENCY EXCHANGE CO PRIVATE LI
Shop no 01 02 Ahmed Building 2 Saddar Cantt
Peshawar, 25000
2. Cross border payments
VI. International Finance Corporation IFC
VII. International Finance Corporation IFC , a World Bank Group company[v]

VIII. Project is being financed by the World Bank (US$588 million)
IX. World Bank okays another $700m for Dasu project [vi]

X. Dasu, Khyber Pakhtunkhwa[vii], Pakistan Hydroelectric plant[viii] map[ix]
XI. A Chinese Company is building the Hydro Electric DAM.
XII. Dasu blockchain mining will participate on Blockchain based Service Network (BSN) , https://bsnbase.io/g/main/index , a Chinese based company. We are building Public City Nodes (PCNs) on all major cloud services' regions or zones. The PCNs are not blockchain nodes, Instead they are virtual data centers with complex blockchain operating environment installed within. All PCNs are connected altogether to form the BSN.
XIII. Technical Stack: blockchain
1. python [x] , goLang
2. Solidity's
3. Securitize[xi]
4. EIP(Ethereum Improvement Proposal)[xii]
5. hydroelectric energy sources[xiii]
6. DeFi, Decentralize Finance
7. DEX, Decentralize Exchange
8. listing on DeFi exchange: Coinbase, binance , bakkt , ICE, etc...
9. UniSwap compound , Solana Serum SOL, San Diego, California[xiv]

XIV. Pakistani province of Khyber Pakhtunkhwa[xv] is planning to build two cryptocurrency mines as part of a pilot program.[xvi]

Purpose of test code project(PoC): Provide a Independent Pakistani Hydro electric producer to securitize a cash flow stream from a Pakistani Hydro electric producer asset to a ERC-20 contract that can be used to extend credit to meet capital needs for operating capitol. Independent Pakistani Hydro electric producer ERC-20 contract address, moves asset value to (=>) ETH Defi address[xvii].

XV. ERC-2222[xviii] [xix](proposed not a standard, not accepted as a ERC20Detailed standard.)(w/code)[1xx]
ERI-2222 — Funds Distribution Standard. A key concept of the ACTUS Protocol[xxi] [xxii] is that financial asset tokens not only represent the ownership of the underlying asset (or it's market value) but, more importantly, the right to participate in the distribution of future payments as defined by the asset terms. Examples of such payments are dividend and interest distributions, the bond repayments or the distribution of payments from pass-through structures as known e.g. from asset-backed securities. Therefore, we have developed in a collaborative effort a funds distribution token standard which is known as ERC-2222. (code attached)

1ERC-2222 code from
Funds Distribution Token (FDT): https://github.com/atpar/funds-distribution-token

XVI. Test the code by transferring asset value on a Local test net. PoC ,Define Independent Pakistani Hydro electric producer address(INOG, PAKH ...etc token name), lookup token on Ether scan and mock GCoin address. Proposed currency exchange Binary Options: USD/PAKH, BTC/PAKH, ETH/PAKH, etc.

XVII. DeFi , Dapp, RoI, Mega Watt rating, HASH Rate, # of Hosts, etc...

RoI	Mega Watt rating/ total@ DASU	#hosts: GPU RTX3080x8	HASH Rate … S19/GPU	# of Hosts Bitmain S19	Live ROI/YoY $USD … S19/GPU
	300/4320	1000		1000	
	0	1	95T/H/760M/H	1	$12K/$31K
		10000	9.5P/H	10000	
		15000	10P/H	15000	

Table 1.

XVIII. Antminer S19, 95.00 T/H, 3215 Watt, 34 W/T, $35.23 24hour Revenue, $2.31 Electricity , 6.57% Electricity ratio, $3867.67 Break even, $32.92 24 Hour profit , $0.03 per kilowatt hour (KWh)

24H Revenue	0.00059850 BTC
Fee/D	0.00003934 BTC
revenue 24H	0.00055916 BTC

Table 2.

04/01/21 : 03:50:47 PM PST USA; $59, 850.00 = 1x BTC , www.fiatleak.com
Miner Profitability Ranking: https://www.poolin.com/miners-rank
$32.92 x 365 days = $12k per year

Price $2014.01 /ETH Other Fees $0 , Per Day Per Miner Network Hashrate: 462.49 T H/s 04/02/21 ; 07:50:43 AM
GPU Miner RTX3080 x 8卡 ,760.00 M/H,
2000 Watts ,2.63 W/M, $88.79 24hr Revenue, $1.44 Electricity, 1.62% Electric Ratio, $32.66 Break even $87.35 24hr profit
https://www.poolin.com/miners-rank
RTX3080 8卡

24H Revenue	0.04408760 ETH
Fee/D	0.00071499 ETH
revenue 24H	0.04337261 ETH

Table 3.
$87.35 x 365 = $31K per year

XIX. Pakistani province of Khyber Pakhtunkhwa is planning to build two cryptocurrency mines as part of a pilot program.

Purpose of test coPakistani province of Khyber Pakhtunkhwa[xxiii] is planning to build two cryptocurrency mines as part of a pilot program.[xxiv]de project(PoC): Provide a Independent Pakistani Hydro electric producer to securitize a cash flow stream from a Pakistani Hydro electric producer asset to a ERC-20 contract that can be used to extend credit to meet capital needs for operating capitol. Independent Pakistani Hydro electric producer ERC-20 contract address, moves asset value to (=>) ETH Defi address.

XX. ERC-2222(proposed not a standard, not accepted as a ERC20Detailed standard.)(w/code)[2]

ERI-2222 — Funds Distribution Standard. A key concept of the ACTUS Protocol is that financial asset tokens not only represent the ownership of the underlying asset (or it's market value) but, more importantly, the right to participate in the distribution of future payments as defined by the asset terms. Examples of such payments are dividend and interest distributions, the bond repayments or the distribution of payments from pass-through structures as known

2ERC-2222 code from
Funds Distribution Token (FDT): https://github.com/atpar/funds-distribution-token

e.g. from asset-backed securities. Therefore, we have developed in a collaborative effort a funds distribution token standard which is known as ERC-2222. (code attached)

XXI. Test the code by transferring asset value on a Local test net. PoC ,Define Independent Pakistani Hydro electric producer address(INOG, PAKH ...etc token name), lookup token on Ether scan and mock GCoin address. Proposed currency exchange Binary Options: USD/PAKH, BTC/PAKH, ETH/PAKH, etc.

meta tags: MoneyGram, P2P, UnBanked, Khyber Pakhtunkhwa, MoneyGram local site, PAKISTAN CURRENCY EXCHANGE CO PRIVATE LI, Shop no 01 02 Ahmed Building 2 Saddar Cantt, Peshawar, 25000, Cross border payments, Pakistani, Khyber Pakhtunkhw, cryptocurrency mines, pilot program, python , goLang, Solidity, Securitize , EIP(Ethereum Improvement Proposal), hydroelectric energy sources , DeFi, Decentralize Finance, DEX, Decentralize Exchange, Coinbase, binance , bakkt , ICE, UniSwap compound , Solana Serum SOL, San Diego, California

i ERC20 is decentralized digital NFTs Collectibles currency. https://erc20.tech

ii TRADING OPTIONS & DERIVATIVES TRADING; Securitization is the procedure where an issuer designs a marketable financial instrument by merging or pooling various financial assets into one group. The issuer then sells this group of repackaged assets to investors. Securitization offers opportunities for investors and frees up capital for originators, both of which promote liquidity in the marketplace. https://www.investopedia.com/terms/s/securitization.asp

iii Chash Flow Plans: Cash Flow Plans: https://www.investopedia.com/terms/c/cash-flow-plans.asp
By WILL KENTON
Updated Nov 20, 2019
What Are Cash Flow Plans?
Cash flow plans, in insurance, are plans that allow policyholders to use their own cash flow to finance their insurance premiums. Cash flow plans can also refer to an insurance company's assessment of a company's cash flow, income streams, and expenses, along with a plan to coordinate the payment of insurance premiums. However, cash flow plans can also relate to documents a company puts together to track cash flow, both cash inflows and outflows, over a period.

iv Zia Ullah Khan Bangash: http://www.pakp.gov.pk/2013/member/pk-38/ :

Committee Chairman Right to Information Committee Member

AGRICULTURE, LIVESTOCK AND COOPERATION DEPARTMENT
DEPARTMENT

Committee ChairmanSTANDING COMMITTEE NO. 15 ON INFORMATION AND PUBLIC RELATIONS DEPARTMENT
Committee ChairmanSTANDING COMMITTEE NO. 03 ON JUDICAL
Committee MemberSTANDING COMMITTEE NO. 35 ON MINERALS DEVELOPMENT DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 22 ON REVENUE DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 18 ON LOCAL GOVERNMENT, ELECTIONS AND RURAL DEVELOPMENT DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 13 ON HOME AND TRIBAL AFFARIS DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 11 ON ENVIRONMENT DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 08 ON HIGHER EDUCATION, ARCHIVES AND LIBRARIES DEPARTMENT
Committee MemberSTANDING COMMITTEE NO. 05 ON AGRICULTURE, LIVESTOCK AND COOPERATION DEPARTMENT
Committee MemberRight to Information

v Project is being financed by the World Bank (US$588 million), the Local Commercial financing from a consortium of Local Banks (Rs144 billion), and Foreign Commercial financing from Credit Suisse Bank ($350 million). WAPDA will also inject its equity equivalent to 15% of the project base cost.

vi ISLAMABAD: The World Bank's board of directors has approved a $700 million additional financing for 250km transmission line of the Dasu Hydropower Project to be completed by December 2023. World Bank okays another $700m for Dasu project

This was announced by the World Bank at a time the government is seeking about $1.2 billion support from the bank through diversion of surplus or unutilised funds from slow-moving projects to meet additional funding requirements to mitigate adverse impacts of coronavirus threat.

vii Wikipedia : https://en.wikipedia.org/wiki/Khyber_Pakhtunkhwa :Khyber Pakhtunkhwa (Pashto: خیبر پښتونخوا; Urdu: خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province, is one of the four provinces of Pakistan. It is located in the northwestern region of the country, along the Afghanistan–Pakistan border.

Khyber Pakhtunkhwa (Pashto : خیبر پښتونخوا; Urdu : خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province , is one of the four provinces of Pakistan . It is located in the northwestern region of the country, along the Afghanistan–Pakistan border .

viii Dasu, Pakistan: hydropower plants under construction, by capacity
Source: Power Technology, International Rivers , Hydropower World Energy Council 2015

The Dasu Dam is a large hydroelectric gravity dam currently under construction on the Indus River near Dasu in Kohistan District, Khyber Pakhtunkhwa Province, Pakistan. It is developed by Pakistan Water and

Power Development Authority (Wapda), as a key component of the company's Water Vision 2025.
https://en.wikipedia.org/wiki/Dasu_Dam
The 242 m (794 ft) tall dam will support a 4,320 MW hydropower station, to be built in two 2,160 MW stages. The plant is expected to start generating power in late 2024, and stage I is planned to complete by early 2025.
Water from the reservoir will be diverted to the power station located about 3.5 km (2.2 mi) downstream.

ix map of Dasu and near places
Google Local — Dasu mapWelcome to the Dasu google satellite map! This place is situated in Kargil, Jammu and Kashmir, Pakistan, its geographical coordinates are 35° 37' 0" North, 75° 19' 0" East and its original name (with diacritics) is Dasu. See Dasu photos and images from satellite below, explore the aerial photographs of Dasu in Pakistan. Dasu hotels map is available on the target page linked above.

x Python programming language : https://www.python.org/about/

xi Asset Backed Securities: http://people.stern.nyu.edu/igiddy/ABS/absprocess.pdf ,
The Securitization Process
Prof. Ian Giddy
Stern School of Business
New York University

xii The Ethereum Improvement Proposals repository exists as a place to share concrete proposals with potential users of the proposal and the Ethereum community at large. https://eips.ethereum.org/ : https://eips.ethereum.org/EIPS/eip-1.

xiii hydroelectric energy sources: Representatives of more than 170 countries reached consensus at the Top World Conference on Sustainable Development, in Johannesburg (2002), and at the 3rd World Forum on Water, in Kyoto (2003): hydroelectric generation is renewable and has certain merits Here are ten reasons leading them to this conclusion.

1. Hydroelectricity is a renewable energy source.

Hydroelectricity uses the energy of running water, without reducing its quantity, to produce electricity. Therefore, all hydroelectric developments, of small or large size, whether run of the river or of accumulated storage, fit the concept of renewable energy.

2. Hydroelectricity makes it feasible to utilize other renewable sources.

Hydroelectric power plants with accumulation reservoirs offer incomparable operational flexibility, since they can immediately respond to fluctuations in the demand for electricity. The flexibility and storage capacity of hydroelectric power plants make them more efficient and economical in supporting the use of intermittent sources of renewable energy, such as solar energy or Aeolian energy.

3. Hydroelectricity promotes guaranteed energy and price stability.

River water is a domestic resource which, contrary to fuel or natural gas, is not subject to market fluctuations. In addition to this, it is the only large renewable source of electricity and its cost-benefit ratio, efficiency, flexibility and reliability assist in optimizing the use of thermal power plants.

4. Hydroelectricity contributes to the storage of drinking water.

Hydroelectric power plant reservoirs collect rainwater, which can then be used for consumption or for irrigation. In storing water, they protect the water tables against depletion and reduce our vulnerability to floods and droughts.

5. Hydroelectricity increases the stability and reliability of electricity systems.

The operation of electricity systems depends on rapid and flexible generation sources to meet peak demands, maintain the system voltage levels, and quickly re-establish supply after a blackout. Energy generated by hydroelectric installations can be injected into the electricity system faster than that of any other energy source. The capacity of hydroelectric systems to reach maximum production from zero in a rapid and foreseeable manner makes them exceptionally appropriate for addressing alterations in the consumption and providing ancillary services to the electricity system, thus maintaining the balance between the electricity supply and demand.

6. Hydroelectricity helps fight climate changes.

The hydroelectric life cycle produces very small amounts of greenhouse gases (GHG). In emitting less GHG than power plants driven by gas, coal or oil, hydroelectricity can help retard global warming. Although only 33% of the available hydroelectric potential has been developed, today hydroelectricity prevents the emission of GHG corresponding to the burning of 4.4 million barrels of petroleum per day worldwide.

7. Hydroelectricity improves the air we breathe.

Hydroelectric power plants don't release pollutants into the air. They very frequently substitute the generation from fossil fuels, thus reducing acid rain and smog. In addition to this, hydroelectric developments don't generate toxic by-products.

8. Hydroelectricity offers a significant contribution to development.

Hydroelectric installations bring electricity, highways, industry and commerce to communities, thus developing the economy, expanding access to health and education, and improving the quality of life. Hydroelectricity is a technology that has been known and proven for more than a century. Its impacts are well understood and manageable through measures for mitigating and compensating the damages. It offers a vast potential and is available where development is most necessary.

9. Hydroelectricity means clean and cheap energy for today and for tomorrow.

With an average lifetime of 50 to 100 years, hydroelectric developments are long-term investments that can benefit various generations. They can be easily upgraded to incorporate more recent technologies and have very low operating and maintenance costs.

10. Hydroelectricity is a fundamental instrument for sustainable development.

Hydroelectric enterprises that are developed and operated in a manner that is economically viable, environmentally sensible and socially responsible represent the best concept of sustainable development. That means, "development that today addresses people's needs without compromising the capacity of future generations for addressing their own needs" (World Commission on the Environment and Development, 1987).

xiv Solana Serum is a decentralized exchange (DEX) and ecosystem that brings unprecedented speed and low transaction costs to decentralized finance. It is built on Solana and is completely permissionless.https://projectserum.com

xv Wikipedia : https://en.wikipedia.org/wiki/Khyber_Pakhtunkhwa :Khyber Pakhtunkhwa (Pashto: خیبر پښتونخوا; Urdu: خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province, is one of the four provinces of Pakistan. It is located in the northwestern region of the country, along the Afghanistan–Pakistan border.

Khyber Pakhtunkhwa (Pashto : خیبر پښتونخوا; Urdu : خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province , is one of the four provinces of Pakistan . It is located in the northwestern region of the country, along the Afghanistan–Pakistan border .

xvi Pakistani province of Khyber Pakhtunkhwa is planning to build two cryptocurrency mines as part of a pilot program. Provincial government in Pakistan plans hydro-powered crypto mining pilot program: report

by Michael McSweeney;

March 18, 2021, 9:52AM EDT · 1 min read

Reuters reported Thursday that the government in the Pakistani province of Khyber Pakhtunkhwa is planning to build two cryptocurrency mines as part of a pilot program.

The pilot mines will draw their power from hydroelectric energy sources, according to the report. According to Reuters, the ultimate cost of the initiative is not yet known.

“People have already been approaching us for investment, and we want them to come to Khyber Pakhtunkhwa, earn some money and have the province earn from that as well,” Zia Ullah Bangash, who advises the provincial government, told Reuters.

The advisor later said: "We are hoping to bring this to a government level so things can be controlled and online fraud or other scams can be prevented."

xvii Swap compound

xviii ERC-2222 - Funds Distribution Standard #2222: https://github.com/ethereum/EIPs/issues/2222#1726

The FDT EIP is a specification for an extension to the ERC20 token interface. It adds functionality to the token that enables it to represent claims on future cash flow of an asset such as dividends, loan repayments, fee or revenue shares. The incoming funds can be distributed efficiently among large numbers of token holders. Anyone can deposit funds, token holders can withdraw their claims.

Abstract

This EIP proposes a standard interface for distributing payments such as dividends, loan repayments, fee or revenue shares among token holders. The token holders are seen as fractional owners of future cash flow. The payments can be in Ether or tokens and are seen as the token's "fund". FDT holders can transfer their tokens at any time and can be sure that their past claims to the cash flow of the token will be honored. The interface provides methods to deposit funds, to get information about available funds and to withdraw funds.

xix Funds Distribution Token (FDT): https://github.com/atpar/funds-distribution-token :

DRAFT IMPLEMENTATION. NOT AUDITED. DO NOT USE FOR TOKENS WITH REAL VALUE AT THIS TIME.

This repository contains a reference implementation of a token using the proposed Funds Distribution Standard, written in solidity.

The Funds Distribution Standard is an extension to the ERC20 token standard that adds the functionality to represent claims on any type of crypto cash flow. It will be submitted as an Ethereum Improvement Proposal (EIP)

xx ERC-2222 code from

Funds Distribution Token (FDT): https://github.com/atpar/funds-distribution-token

I. ===/==FundsDistributionToken.sol==/===

```
1. pragma solidity ^0.5.2;
2.
3. import "openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol";
4. import "openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol";
5. import "openzeppelin-solidity/contracts/math/SafeMath.sol";
6. import "./external/math/SafeMathUint.sol";
7. import "./external/math/SafeMathInt.sol";
8.
9. import "./IFundsDistributionToken.sol";
10.
11.
12. /**
13.  * @title FundsDistributionToken
14.  * @author Johannes Escherich
15.  * @author Roger-Wu
16.  * @author Johannes Pfeffer
17.  * @author Tom Lam
18.  * @dev A  mintable token that can represent claims on cash flow of arbitrary assets such as dividends, loan repayments,
19.  * fee or revenue shares among large numbers of token holders. Anyone can deposit funds, token holders can withdraw
20.  * their claims.
21.  * FundsDistributionToken (FDT) implements the accounting logic. FDT-Extension contracts implement methods for depositing and
22.  * withdrawing funds in Ether or according to a token standard such as ERC20, ERC223, ERC777.
23.  */
24. contract FundsDistributionToken is IFundsDistributionToken, ERC20Detailed, ERC20Mintable {
25.
```

```
26.         using SafeMath for uint256;
27.         using SafeMathUint for uint256;
28.         using SafeMathInt for int256;
29.
30.         // optimize, see
    https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
31.         uint256 constant internal pointsMultiplier = 2**128;
32.         uint256 internal pointsPerShare;
33.
34.         mapping(address => int256) internal pointsCorrection;
35.         mapping(address => uint256) internal withdrawnFunds;
36.
37.
38.         constructor (
39.                 string memory name,
40.                 string memory symbol
41.         )
42.                 public
43.                 ERC20Detailed(name, symbol, 18)
44.         {}
45.
46.         /**
47.          * prev. distributeDividends
48.          * @notice Distributes funds to token holders.
49.          * @dev It reverts if the total supply of tokens is 0.
50.          * It emits the `FundsDistributed` event if the amount of received ether is
    greater than 0.
51.          * About undistributed funds:
52.          *   In each distribution, there is a small amount of funds which does not
    get distributed,
53.          *       which is `(msg.value * pointsMultiplier) % totalSupply()`.
54.          *   With a well-chosen `pointsMultiplier`, the amount funds that are not
    getting distributed
55.          *       in a distribution can be less than 1 (base unit).
56.          *   We can actually keep track of the undistributed ether in a distribution
57.          *       and try to distribute it in the next distribution ..... todo implement
58.          */
59.         function _distributeFunds(uint256 value) internal {
60.                 require(totalSupply() > 0, "FundsDistributionToken._distributeFunds:
    SUPPLY_IS_ZERO");
61.
62.                 if (value > 0) {
63.                         pointsPerShare = pointsPerShare.add(
64.                                 value.mul(pointsMultiplier) / totalSupply()
65.                         );
66.                         emit FundsDistributed(msg.sender, value);
67.                 }
68.         }
69.
70.         /**
71.          * prev. withdrawDividend
```

```
72.          * @notice Prepares funds withdrawal
73.          * @dev It emits a `FundsWithdrawn` event if the amount of withdrawn
    ether is greater than 0.
74.          */
75.         function _prepareWithdraw() internal returns (uint256) {
76.                 uint256 _withdrawableDividend = withdrawableFundsOf(msg.sender);
77.
78.                 withdrawnFunds[msg.sender] =
    withdrawnFunds[msg.sender].add(_withdrawableDividend);
79.
80.                 emit FundsWithdrawn(msg.sender, _withdrawableDividend);
81.
82.                 return _withdrawableDividend;
83.         }
84.
85.         /**
86.          * prev. withdrawableDividendOf
87.          * @notice View the amount of funds that an address can withdraw.
88.          * @param _owner The address of a token holder.
89.          * @return The amount funds that `_owner` can withdraw.
90.          */
91.         function withdrawableFundsOf(address _owner) public view returns(uint256)
    {
92.                 return accumulativeFundsOf(_owner).sub(withdrawnFunds[_owner]);
93.         }
94.
95.         /**
96.          * prev. withdrawnDividendOf
97.          * @notice View the amount of funds that an address has withdrawn.
98.          * @param _owner The address of a token holder.
99.          * @return The amount of funds that `_owner` has withdrawn.
100.                 */
101.                 function withdrawnFundsOf(address _owner) public view
    returns(uint256) {
102.                         return withdrawnFunds[_owner];
103.                 }
104.
105.                 /**
106.                  * prev. accumulativeDividendOf
107.                  * @notice View the amount of funds that an address has earned in
    total.
108.                  * @dev accumulativeFundsOf(_owner) =
    withdrawableFundsOf(_owner) + withdrawnFundsOf(_owner)
109.                  * = (pointsPerShare * balanceOf(_owner) +
    pointsCorrection[_owner]) / pointsMultiplier
110.                  * @param _owner The address of a token holder.
111.                  * @return The amount of funds that `_owner` has earned in total.
112.                  */
113.                 function accumulativeFundsOf(address _owner) public view
    returns(uint256) {
114.                         return pointsPerShare.mul(balanceOf(_owner)).toInt256Safe()
```

```
115.                                 .add(pointsCorrection[_owner]).toUint256Safe() /
    pointsMultiplier;
116.             }
117.
118.             /**
119.              * @dev Internal function that transfer tokens from one address to
    another.
120.              * Update pointsCorrection to keep funds unchanged.
121.              * @param from The address to transfer from.
122.              * @param to The address to transfer to.
123.              * @param value The amount to be transferred.
124.              */
125.             function _transfer(address from, address to, uint256 value) internal {
126.                     super._transfer(from, to, value);
127.
128.                     int256 _magCorrection =
    pointsPerShare.mul(value).toInt256Safe();
129.                     pointsCorrection[from] =
    pointsCorrection[from].add(_magCorrection);
130.                     pointsCorrection[to] =
    pointsCorrection[to].sub(_magCorrection);
131.             }
132.
133.             /**
134.              * @dev Internal function that mints tokens to an account.
135.              * Update pointsCorrection to keep funds unchanged.
136.              * @param account The account that will receive the created
    tokens.
137.              * @param value The amount that will be created.
138.              */
139.             function _mint(address account, uint256 value) internal {
140.                     super._mint(account, value);
141.
142.                     pointsCorrection[account] = pointsCorrection[account]
143.                             .sub( (pointsPerShare.mul(value)).toInt256Safe() );
144.             }
145.
146.             /**
147.              * @dev Internal function that burns an amount of the token of a
    given account.
148.              * Update pointsCorrection to keep funds unchanged.
149.              * @param account The account whose tokens will be burnt.
150.              * @param value The amount that will be burnt.
151.              */
152.             function _burn(address account, uint256 value) internal {
153.                     super._burn(account, value);
154.
155.                     pointsCorrection[account] = pointsCorrection[account]
156.                             .add( (pointsPerShare.mul(value)).toInt256Safe() );
157.             }
158.     }
```

II. ===/==IFundsDistributionToken.sol==/===

```
1.  pragma solidity ^0.5.2;
2.
3.
4.  interface IFundsDistributionToken {
5.
6.          /**
7.           * @dev Returns the total amount of funds a given address is able to
    withdraw currently.
8.           * @param owner Address of FundsDistributionToken holder
9.           * @return A uint256 representing the available funds for a given account
10.          */
11.         function withdrawableFundsOf(address owner) external view returns
    (uint256);
12.
13.         /**
14.          * @dev Withdraws all available funds for a FundsDistributionToken holder.
15.          */
16.         function withdrawFunds() external;
17.
18.         /**
19.          * @dev This event emits when new funds are distributed
20.          * @param by the address of the sender who distributed funds
21.          * @param fundsDistributed the amount of funds received for distribution
22.          */
23.         event FundsDistributed(address indexed by, uint256 fundsDistributed);
24.
25.         /**
26.          * @dev This event emits when distributed funds are withdrawn by a token
    holder.
27.          * @param by the address of the receiver of funds
28.          * @param fundsWithdrawn the amount of funds that were withdrawn
29.          */
30.         event FundsWithdrawn(address indexed by, uint256 fundsWithdrawn);
31. }
```

III. ===/==IFundsDistributionTokenOptional.sol==/===

```
1.  pragma solidity ^0.5.2;
2.
3.  import "openzeppelin-solidity/contracts/token/ERC20/IERC20.sol";
4.
5.
6.  interface IFundsDistributionTokenOptional {
7.
8.          /**
9.           * @notice Deposits funds to this contract.
10.          * The deposited funds may be distributed to other accounts.
11.          */
12.         function depositFunds() external payable;
13.
14.         /**
```

```
     * @notice Returns the total amount of funds that have been deposited to this contract but not yet distributed.
     */
    function undistributedFunds() external view returns(uint256);

    /**
     * @notice Returns the total amount of funds that have been distributed.
     */
    function distributedFunds() external view returns(uint256);

    /**
     * @notice Distributes undistributed funds to accounts.
     */
    function distributeFunds() external;

    /**
     * @notice Deposits and distributes funds to accounts.
     * @param from The source of the funds.
     */
    function depositAndDistributeFunds(address from) external payable;

    /**
     * @dev This event MUST emit when funds are deposited to this contract.
     * @param by the address of the sender of who deposited funds.
     * @param fundsDeposited The amount of distributed funds.
     */
    event FundsDeposited(address indexed by, uint256 fundsDeposited);
}
```

IV. ===/==FDT_ERC20Extension.sol==/===

```
ragma solidity ^0.5.2;

import "../external/math/SafeMathUint.sol";
import "../external/math/SafeMathInt.sol";

import "../IFundsDistributionToken.sol";
import "../FundsDistributionToken.sol";


contract FDT_ERC20Extension is IFundsDistributionToken, FundsDistributionToken {

    using SafeMathUint for uint256;
    using SafeMathInt for int256;

    // token in which the funds can be sent to the FundsDistributionToken
    IERC20 public fundsToken;

```

```
18.         // balance of fundsToken that the FundsDistributionToken currently
    holds
19.         uint256 public fundsTokenBalance;
20.
21.
22.         modifier onlyFundsToken () {
23.                 require(msg.sender == address(fundsToken),
    "FDT_ERC20Extension.onlyFundsToken: UNAUTHORIZED_SENDER");
24.                 _;
25.         }
26.
27.         constructor(
28.                 string memory name,
29.                 string memory symbol,
30.                 IERC20 _fundsToken
31.         )
32.                 public
33.                 FundsDistributionToken(name, symbol)
34.         {
35.                 require(address(_fundsToken) != address(0),
    "FDT_ERC20Extension: INVALID_FUNDS_TOKEN_ADDRESS");
36.
37.                 fundsToken = _fundsToken;
38.         }
39.
40.         /**
41.          * @notice Withdraws all available funds for a token holder
42.          */
43.         function withdrawFunds()
44.                 external
45.         {
46.                 uint256 withdrawableFunds = _prepareWithdraw();
47.
48.                 require(fundsToken.transfer(msg.sender, withdrawableFunds),
    "FDT_ERC20Extension.withdrawFunds: TRANSFER_FAILED");
49.
50.                 _updateFundsTokenBalance();
51.         }
52.
53.         /**
54.          * @dev Updates the current funds token balance
55.          * and returns the difference of new and previous funds token
    balances
56.          * @return A int256 representing the difference of the new and
    previous funds token balance
57.          */
58.         function _updateFundsTokenBalance() internal returns (int256) {
59.                 uint256 prevFundsTokenBalance = fundsTokenBalance;
60.
61.                 fundsTokenBalance = fundsToken.balanceOf(address(this));
62.
```

```
63.                 return
    int256(fundsTokenBalance).sub(int256(prevFundsTokenBalance));
64.         }
65.
66.         /**
67.          * @notice Register a payment of funds in tokens. May be called
    directly after a deposit is made.
68.          * @dev Calls _updateFundsTokenBalance(), whereby the contract
    computes the delta of the previous and the new
69.          * funds token balance and increments the total received funds
    (cumulative) by delta by calling _registerFunds()
70.          */
71.         function updateFundsReceived() external {
72.                 int256 newFunds = _updateFundsTokenBalance();
73.
74.                 if (newFunds > 0) {
75.                         _distributeFunds(newFunds.toUint256Safe());
76.                 }
77.         }
78. }
```

V. ===/===FDT_ERC223Extension.sol==/===

```
1.      pragma solidity ^0.5.2;
2.
3.      import "../IFundsDistributionToken.sol";
4.      import "../FundsDistributionToken.sol";
5.
6.
7.      contract FDT_ERC223Extension is IFundsDistributionToken,
FundsDistributionToken {
8.
9.              // token that ClaimsToken takes in custodianship
10.             IERC20 public fundsToken;
11.
12.             modifier onlyFundsToken () {
13.                     require(msg.sender == address(fundsToken),
"FDT_ERC223Extension.onlyFundsToken: UNAUTHORIZED_SENDER");
14.                     _;
15.             }
16.
17.             constructor(
18.                     string memory name,
19.                     string memory symbol,
20.                     IERC20 _fundsToken
21.             )
22.                     public
23.                     FundsDistributionToken(name, symbol)
24.             {
25.                     require(address(_fundsToken) != address(0),
"FDT_ERC223Extension: INVALID_FUNDS_TOKEN_ADDRESS");
26.
```

```
27.                     fundsToken = _fundsToken;
28.             }
29.
30.             /**
31.              * @dev Withdraws available funds for user.
32.              */
33.             function withdrawFunds()
34.                     external
35.             {
36.                     uint256 withdrawableFunds = _prepareWithdraw();
37.
38.                     require(fundsToken.transfer(msg.sender, withdrawableFunds),
"FDT_ERC223Extension.withdrawFunds: TRANSFER_FAILED");
39.             }
40.
41.             /**
42.              * @dev For ERC223.
43.              * Calls _registerFunds(), whereby magnifiedFundsPerShare gets
updated.
44.              * @param sender Sender of tokens
45.              * @param value Amount of tokens
46.              */
47.             function tokenFallback(address sender, uint256 value, bytes memory)
48.                     public
49.                     onlyFundsToken()
50.             {
51.                     if (value > 0) {
52.                             _distributeFunds(value);
53.                             emit FundsDistributed(sender, value);
54.                     }
55.             }
56.     }
```

VI. ====/===FDT_ETHExtension.sol==/===

```
1.  pragma solidity ^0.5.2;
2.
3.  import "../IFundsDistributionToken.sol";
4.  import "../FundsDistributionToken.sol";
5.
6.
7.  contract FDT_ETHExtension is IFundsDistributionToken,
    FundsDistributionToken {
8.
9.          constructor(
10.                 string memory name,
11.                 string memory symbol
12.         )
13.                 public
14.                 FundsDistributionToken(name, symbol)
15.         {}
16.
```

```
        /**
         * @notice Withdraws available funds for user.
         */
        function withdrawFunds()
                external
        {
                uint256 withdrawableFunds = _prepareWithdraw();

                msg.sender.transfer(withdrawableFunds);
        }

        /**
         * @notice The default function calls _distributeFunds() whereby magnifiedFundsPerShare gets updated.
         */
        function ()
                external
                payable
        {
                if (msg.value > 0) {
                        _distributeFunds(msg.value);
                        emit FundsDistributed(msg.sender, msg.value);
                }
        }
}
```

VII. ===/==ERC20SampleToken.sol==/==

```
pragma solidity ^0.5.2;

import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";


contract ERC223ReceivingContract {
  function tokenFallback(address _from, uint _value, bytes memory _data) public;
}

contract ERC20SampleToken is ERC20 {

        uint8 public constant DECIMALS = 18;
        uint256 public constant INITIAL_SUPPLY = 10000 * (10 ** uint256(DECIMALS));


        constructor () public {
                _mint(msg.sender, INITIAL_SUPPLY);
        }
}
```

VIII. ===/==ERC223SampleToken.sol===/===

```
pragma solidity ^0.5.2;

```

```
3.  import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";
4.
5.
6.  contract ERC223ReceivingContract {
7.    function tokenFallback(address _from, uint _value, bytes memory
    _data) public;
8.  }
9.
10. contract ERC223SampleToken is ERC20 {
11.
12.         uint8 public constant DECIMALS = 18;
13.         uint256 public constant INITIAL_SUPPLY = 10000 * (10 **
    uint256(DECIMALS));
14.
15.
16.         constructor () public {
17.                 _mint(msg.sender, INITIAL_SUPPLY);
18.         }
19.
20.         function transfer(address _to, uint256 _value, bytes memory
    _data)
21.                 public
22.                 returns(bool)
23.         {
24.                 uint codeLength;
25.
26.                 assembly { codeLength := extcodesize(_to) }
27.
28.                 require(super.transfer(_to, _value),
    "TRANFER_FAILED");
29.
30.                 if(codeLength > 0) {
31.                         ERC223ReceivingContract receiver =
    ERC223ReceivingContract(_to);
32.                         receiver.tokenFallback(msg.sender, _value,
    _data);
33.                 }
34.
35.                 return true;
36.         }
37.
38.         function transfer(address _to, uint256 _value)
39.                 public
40.                 returns(bool)
41.         {
42.                 uint codeLength;
43.                 bytes memory empty;
44.
45.                 assembly { codeLength := extcodesize(_to) }
46.
```

```
47.                 require(super.transfer(_to, _value),
    "TRANFER_FAILED");
48.
49.                 if(codeLength > 0) {
50.                         ERC223ReceivingContract receiver =
    ERC223ReceivingContract(_to);
51.                         receiver.tokenFallback(msg.sender, _value,
    empty);
52.                 }
53.
54.                 return true;
55.         }
56. }
```

xxi ACTUS Protocol is a smart contract system for issuing and servicing financial assets on Ethereum.
ACTUS FOUNDATION: https://www.actusfrf.org/about : ALGORITHMIC CONTRACT TYPES UNIFIED STANDARDS
Project ACTUS has been incorporated in the form of two not-for-profit corporations to hold the intellectual property of ACTUS and to further the development as well as maintenance of the ACTUS Contract Types and their use. The not-for-profit corporations with legal residence in the US are the following.© 2019. ACTUS Financial Research Foundation contact@actusfrf.org, 1507 Blue Meadow Road, Rockville, MD 20854, USA

xxii ACTUS protocol:

A C T U S

(1)ACTUS Protocol Portal
(2)Introduction - ACTUS Protocol
(6)Issue and service a Loan - ACTUS Protocol
(7)Tokenize an asset - ACTUS Protocol
ACTUS Technical Specification
actus-solidity-tool
Announcing the ACTUS Protocol Portal | by Nils Bundi | atpar blog | Medium
API v1 (Public Documentation)
API v1 2222(Public Documentation)
atpar ACTUS
atpar blog – Medium
Data Dictionary | ACTUS
demo ACTUS App
demo COM ACTUS App
Developer Dashboard
GitHub - actusfrf/actus-dictionary: This repository contains the actus dictionary and generation scripts
GitHub - atpar/actus-solidity: Solidity implementation of ACTUS Contract Types [MOVED]
GitHub - atpar/ap-monorepo: Monorepo containing all packages related to the ACTUS Protocol
Home | ACTUS
How to use the ACTUS Protocol Portal | by Nils Bundi | atpar blog | Medium
New DLT Law in Switzerland. The best links to learn more. | by Michael Svoboda | atpar blog | Feb, 2021 | Medium
Package actus-solidity · atpar/ap-monorepo · GitHub
Taxonomy | ACTUS
Technical Specification | ACTUS

xxiii Wikipedia : https://en.wikipedia.org/wiki/Khyber_Pakhtunkhwa :Khyber Pakhtunkhwa (Pashto: خیبر پښتونخوا; Urdu: خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province, is one of the four provinces of Pakistan. It is located in the northwestern region of the country, along the Afghanistan–Pakistan border.

Khyber Pakhtunkhwa (Pashto : خیبر پښتونخوا; Urdu : خیبر پختونخوا),[1] often abbreviated as KP or KPK and formerly known as the North-West Frontier Province , is one of the four provinces of Pakistan . It is located in the northwestern region of the country, along the Afghanistan–Pakistan border .

xxiv Pakistani province of Khyber Pakhtunkhwa is planning to build two cryptocurrency mines as part of a pilot program. Provincial government in Pakistan plans hydro-powered crypto mining pilot program: report

by Michael McSweeney;
March 18, 2021, 9:52AM EDT · 1 min read

Reuters reported Thursday that the government in the Pakistani province of Khyber Pakhtunkhwa is planning to build two cryptocurrency mines as part of a pilot program.

The pilot mines will draw their power from hydroelectric energy sources, according to the report. According to Reuters, the ultimate cost of the initiative is not yet known.

“People have already been approaching us for investment, and we want them to come to Khyber Pakhtunkhwa, earn some money and have the province earn from that as well,” Zia Ullah Bangash, who advises the provincial government, told Reuters.

The advisor later said: "We are hoping to bring this to a government level so things can be controlled and online fraud or other scams can be prevented."