Next Generation

Decentralized Intelligence Exchange

TIEX Protocol

2/27/2023

Next Intelligence Exchange

Overview

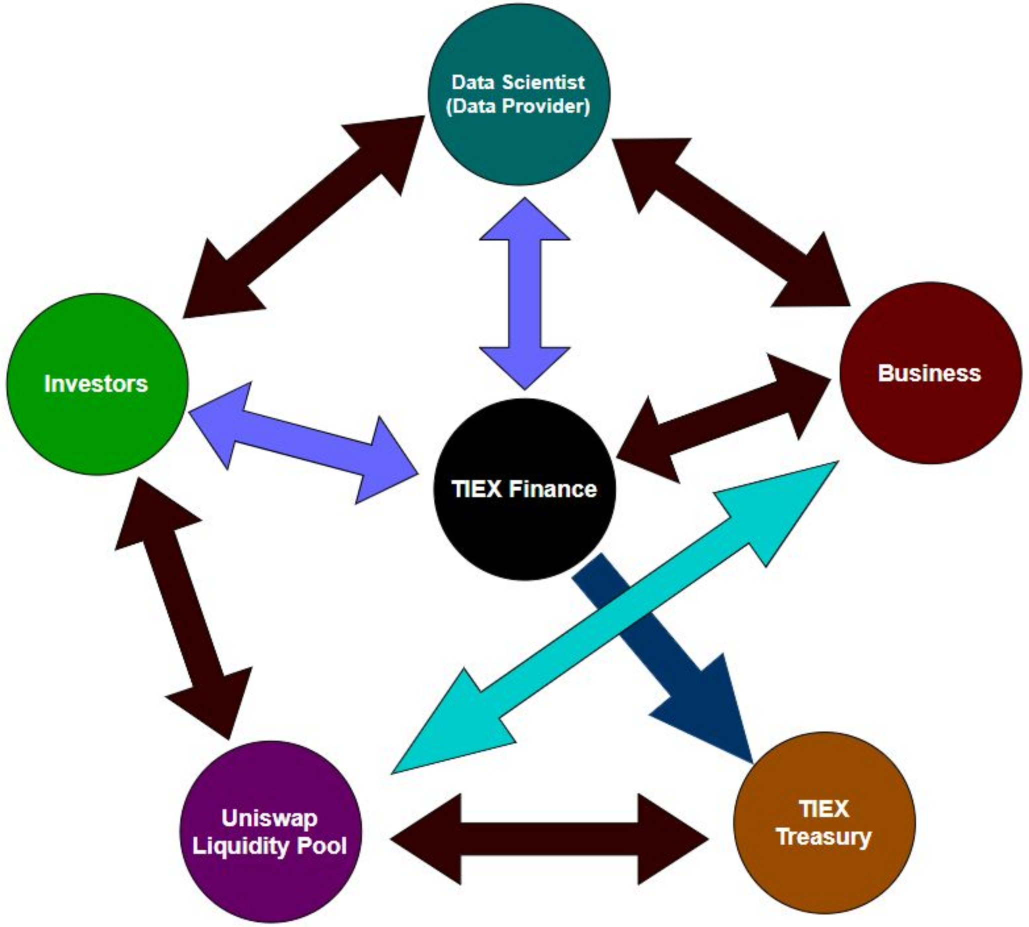
The **TIEX** project is a blockchain-based next-generation decentralized intelligence exchange Protocol that can easily publish large-scale data such as artificial intelligence and machine learning, enable easy access and use by consumers, and provide new investment opportunities to investors.

On the **TIEX** Market, data is published as an interoperable ERC721 data NFT, and data purchase, sale, consumption, and transactions are recorded on the blockchain to make it impossible to forge. And data scientists or AI practitioners will benefit from access to more data, cryptographically secure sources of data and AI Train, income opportunities for data sales and data curating.

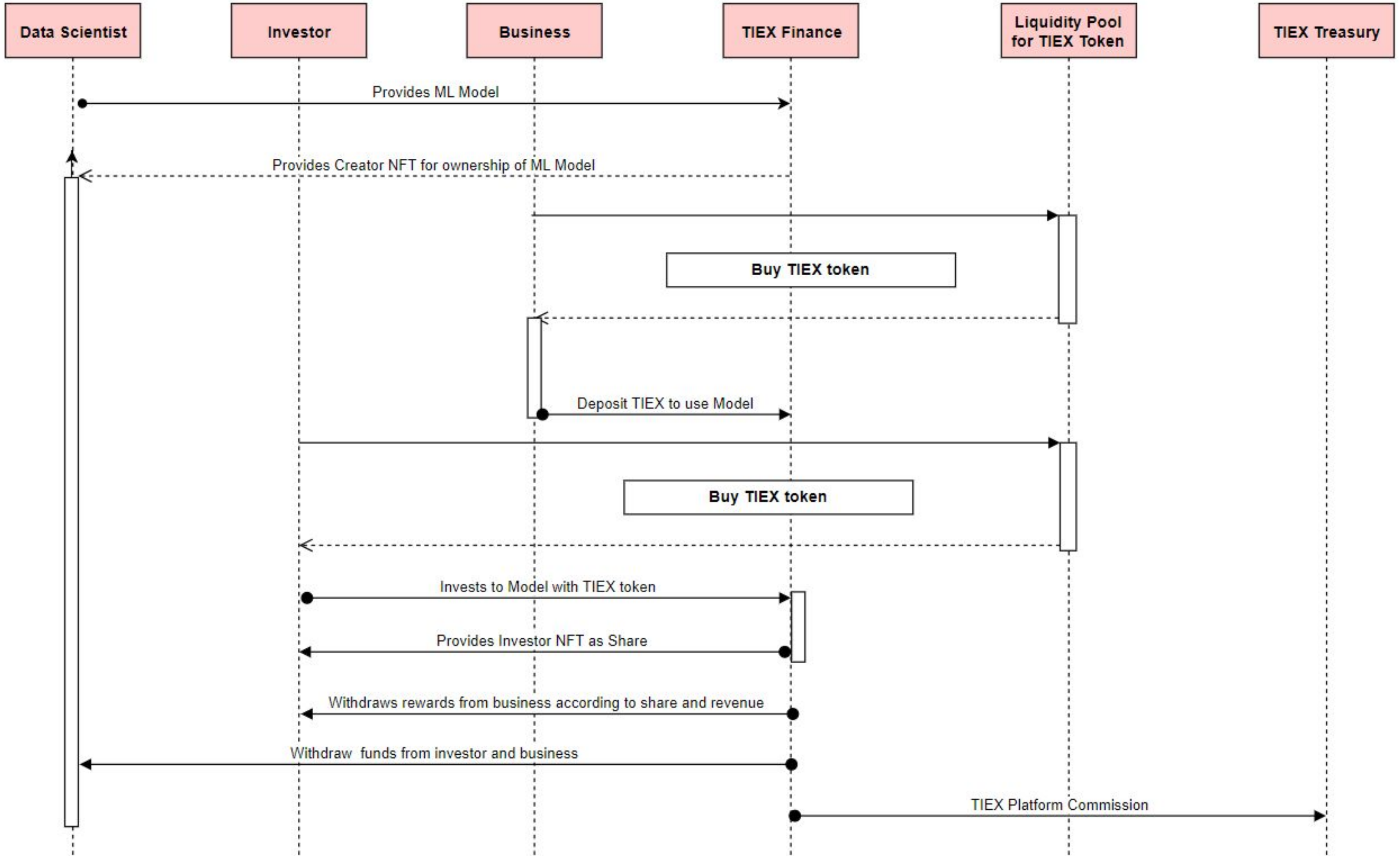
Consumers can access data by using tiex.py sdk library provided by us.

What can they do with TIEX?

1. The Data Scientists can publish their Data on TIEX Protocol and access it with ERC721 Data NFT that we provide.
2. Data that is provided by Data Scientist is used by consumers and consumers can have easy, direct access to resources that are relevant and helpful to their business through our Protocol.
3. The Data scientists can get the funds from investors on the TIEX Protocol, depending on the requirements, and when the revenue of data model reaches the milestones and duration defined by TIEX, the data scientists can finally access and withdraw the investment fund.
4. The investors have the opportunity to invest in data models that are valuable, have significant in the future, and will benefit global enterprises. The investors get more profits when the demand for data is greater. The profit is permanently provided as long as the data model exists in the TIEX Protocol.
5. The data scientists can sell their data models to other data scientists or investors through the TIEX Marketplace as needed.
6. The investors can resell Investor License NFTs received from data scientists (owners of data models) to other investors through the TIEX Marketplace as needed.

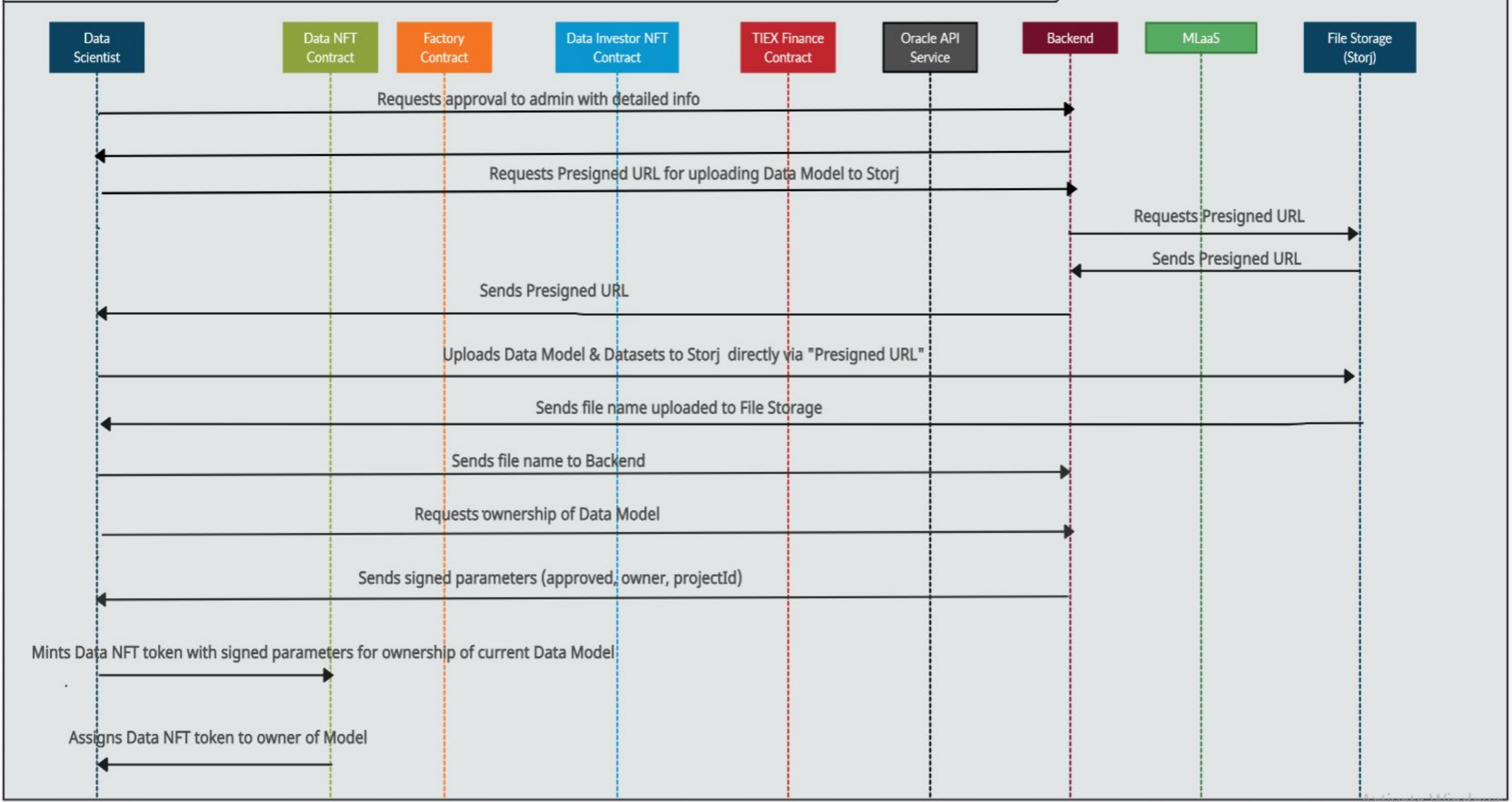


TIEX Finance Lifecycle


Data Scientist
Investor
Business
TIEX Finance
Liquidity Pool for TIEX Token
TIEX Treasury
Provides ML Model
Provides Creator NFT for ownership of ML Model
Buy TIEX token
Deposit TIEX to use Model
Buy TIEX token
Invests to Model with TIEX token
Provides Investor NFT as Share
Withdraws rewards from business according to share and revenue
Withdraw funds from investor and business
TIEX Platform Commission

TIEX Finance Life Cycle Sequence Diagram

1. Sequence Diagram for Data Model Submission & Ownership Assignment


Data Scientist
Data NFT Contract
Factory Contract
Data Investor NFT Contract
TIEX Finance Contract
Oracle API Service
Backend
MLaaS
File Storage (Storj)
Requests approval to admin with detailed info
Requests Presigned URL for uploading Data Model to Storj
Requests Presigned URL
Sends Presigned URL
Sends Presigned URL
Uploads Data Model & Datasets to Storj directly via "Presigned URL"
Sends file name uploaded to File Storage
Sends file name to Backend
Requests ownership of Data Model
Sends signed parameters (approved, owner, projectId)
Mints Data NFT token with signed parameters for ownership of current Data Model
Assigns Data NFT token to owner of Model

Smart Contracts & Oracles For TIEX Protocol

1. TIEX Utility Token Contract for payment token **[Completed]**
2. ERC721 Data NFT Contract **[Completed]**
3. ERC721 Investor License NFT Contract **[Pending]**
4. Factory Contract for generating Investor License NFT Contract **[Pending]**
5. Data NFT Marketplace Contract **[Not Started Yet]**
6. Investor License NFT Marketplace Contract **[Not Started Yet]**
7. TIEX Finance Contract **[Not Started Yet]**
8. Oracle Data Provider **[Not Started Yet]**
9. Oracle Data Consumer Contract **[Not Started Yet]**

1. TIEX Utility Token Contract for payment token

It is Utility Token for a payment that is used for accessing the data model on the TIEX Market.

2. ERC721 Data NFT Contract

This is a non-fungible token (NFT) stored on the blockchain, represents holding copyright/base IP of data model assets and providing the highest authority for data model assets.

Also, it can transfer ownership of copyright/Base IP through transfer transaction. You can also sell ownership of Data Model assets on the Marketplace.

3. ERC721 Investor License NFT Contract

This token is a non-fungible token stored in the blockchain and represents the possession of a license as an investor in Data Model assets, and can have the share in the revenue generated by Data Model assets. In addition, you can sell license ownership as a data model asset investor on the TIEX Marketplace.

ERC721 Investor License NFT Contract is created by Factory Contract from Data NFT token holder.

4. Factory Contract

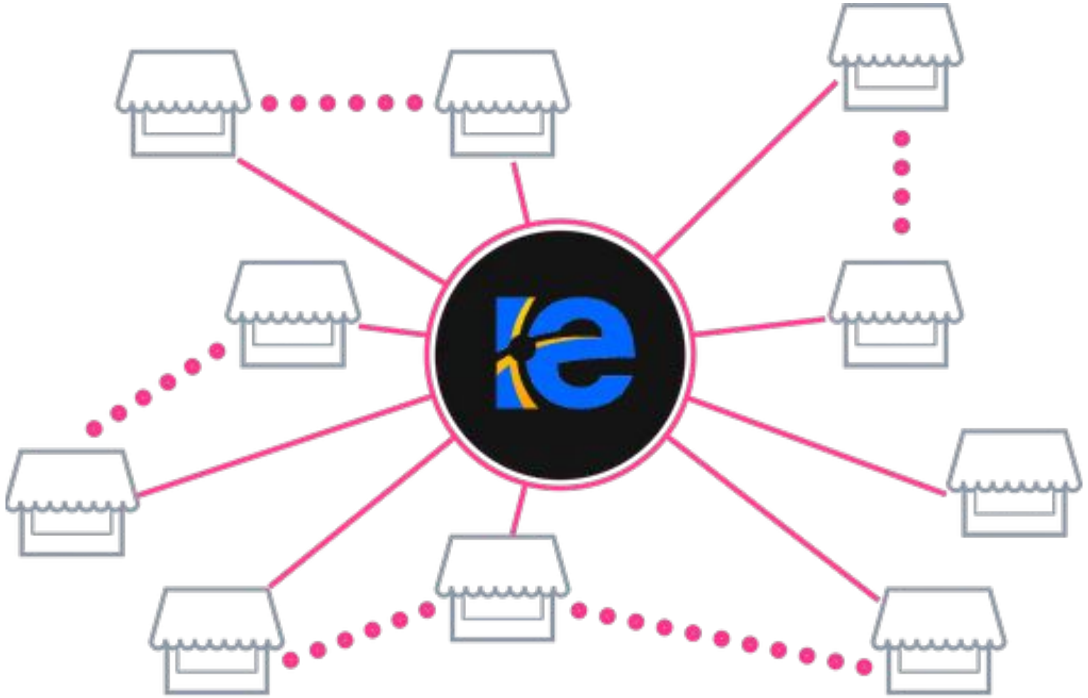
This Smart Contract generates a different ERC721 Investor License NFT Contract for each Data Model when a data scientist (Data NFT token holder) wants investors.

Data NFT token holders provide the following the **parameters [Table 1 - 1] below** when generating the ERC721 Investor License NFT contract through the Factory Contract.

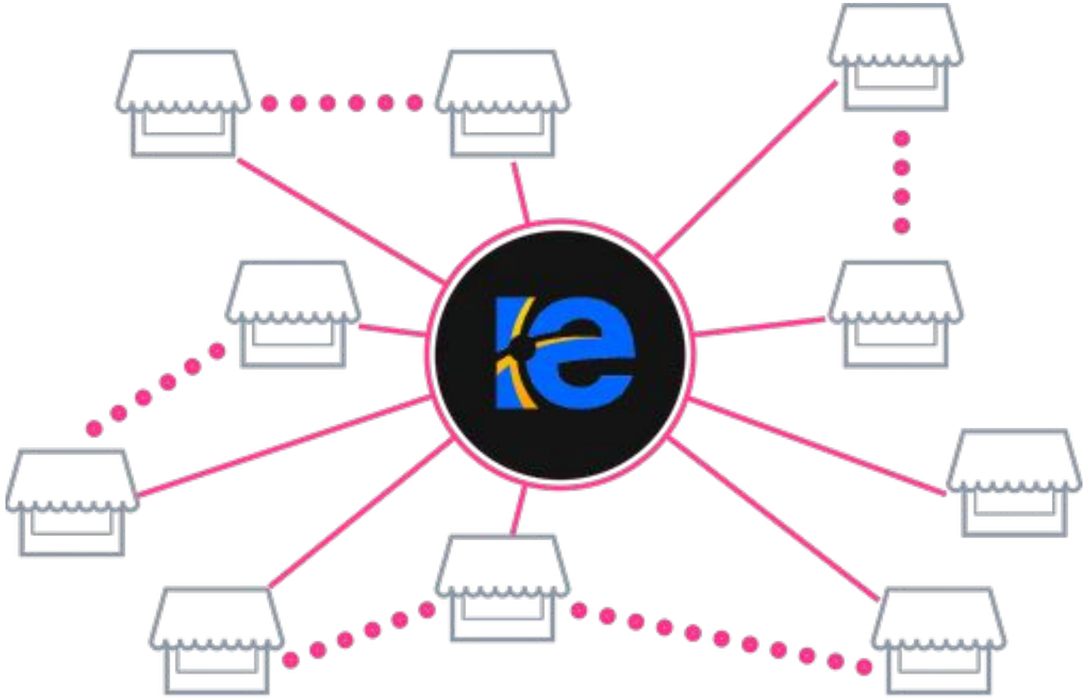
Type	Description	Variable
uint256	Data NFT Token ID	DATA_NFT_TOKEN_ID
address	Data NFT Token Contract Address	DATA_NFT_ADDR
address	TIEX token address	TOKEN_ADDR
string	Name for Data Investor License NFT Token	NAME
string	Symbol for Data Investor License NFT Token	SYMBOL
uint256	Maximum total supply	MAX_TOTAL_SUPPLY
uint256	Price per a share	MINT_PRICE
uint256	Maximum number of shares per an wallet	MAX_MINT_PER_ADDR
uint256	Maximum number of shares per an tx	MAX_MINT_PER_TX
uint256	Launch start time	LAUNCH_START_TIME
uint256	Launch end time	LAUNCH_END_TIME

Table 1-1

5. Data NFT Marketplace Contract



6. Investor License NFT Marketplace



7. Oracle Data Provider [Figure 1-1 Below]

Backend (off-chain database) and Chain Link Server are combined in order for the Chain Link Server to provide parameters such as the number of calls, revenue of the Data Model and claim amount of investor with Oracle Data Consumer.

[Figure 1-1]

8. Oracle Data Consumer [Figure 1-1 Below]

The parameters such as the number of calls, revenue of the Data Model and claim amount of investor provided by Backend is from Oracle Data Provider through Chain Link Server.

[Figure 1-1]

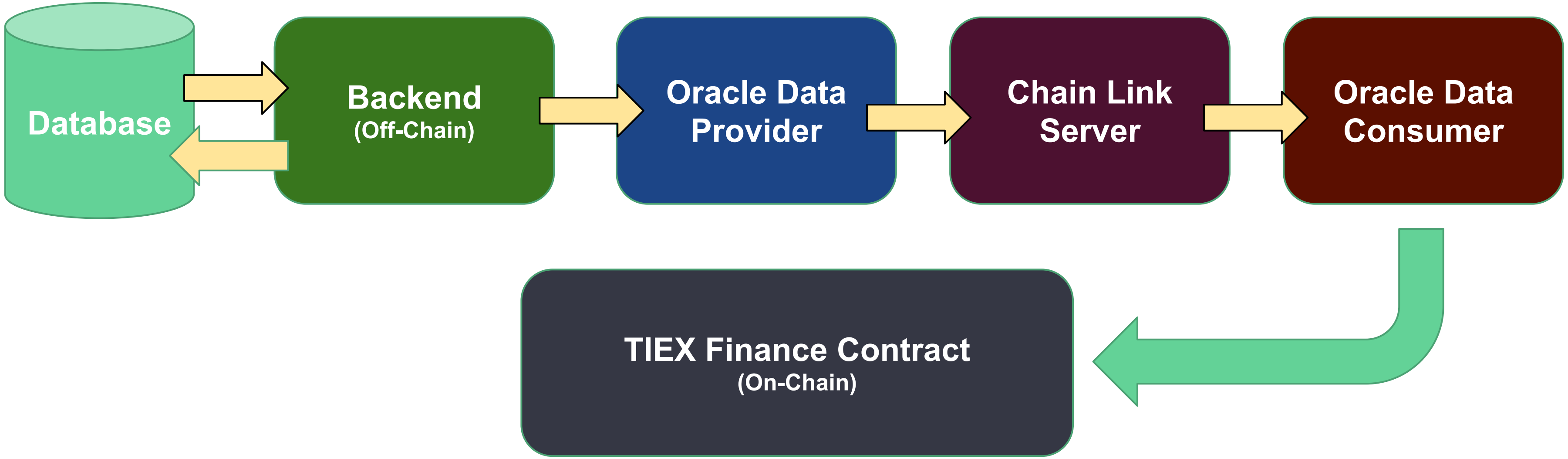


Figure 1-1 Integration off-chain with on-chain for TIEX Finance

9. TIEX Finance Smart Contract   [Figure 1-1 Above]

It gets off-chain information such as number of calls, revenue, and amount of claims from Oracle Data Consumer.

❖ Data consumers first deposit TIEX tokens through **TIEX Finance Contract** to access and use the Data Model in the TIEX Protocol. They withdraw the remaining TIEX tokens through TIEX Finance Contract at any time.
❖ The investors (Data Investor License NFT holders) claim TIEX Tokens through **TIEX Finance Contract** according to the Revenue of the data model and Share of investor.

[Figure 1-1 Above]

High level of structure & Business flow

https://www.figma.com/file/g2lGML9uyXIVPSxbtX7lWQ/Watermarked-ML-Model-NFT-Marketplace?node-id=894%3A66&t=iStWqEco0lfksWAL-0

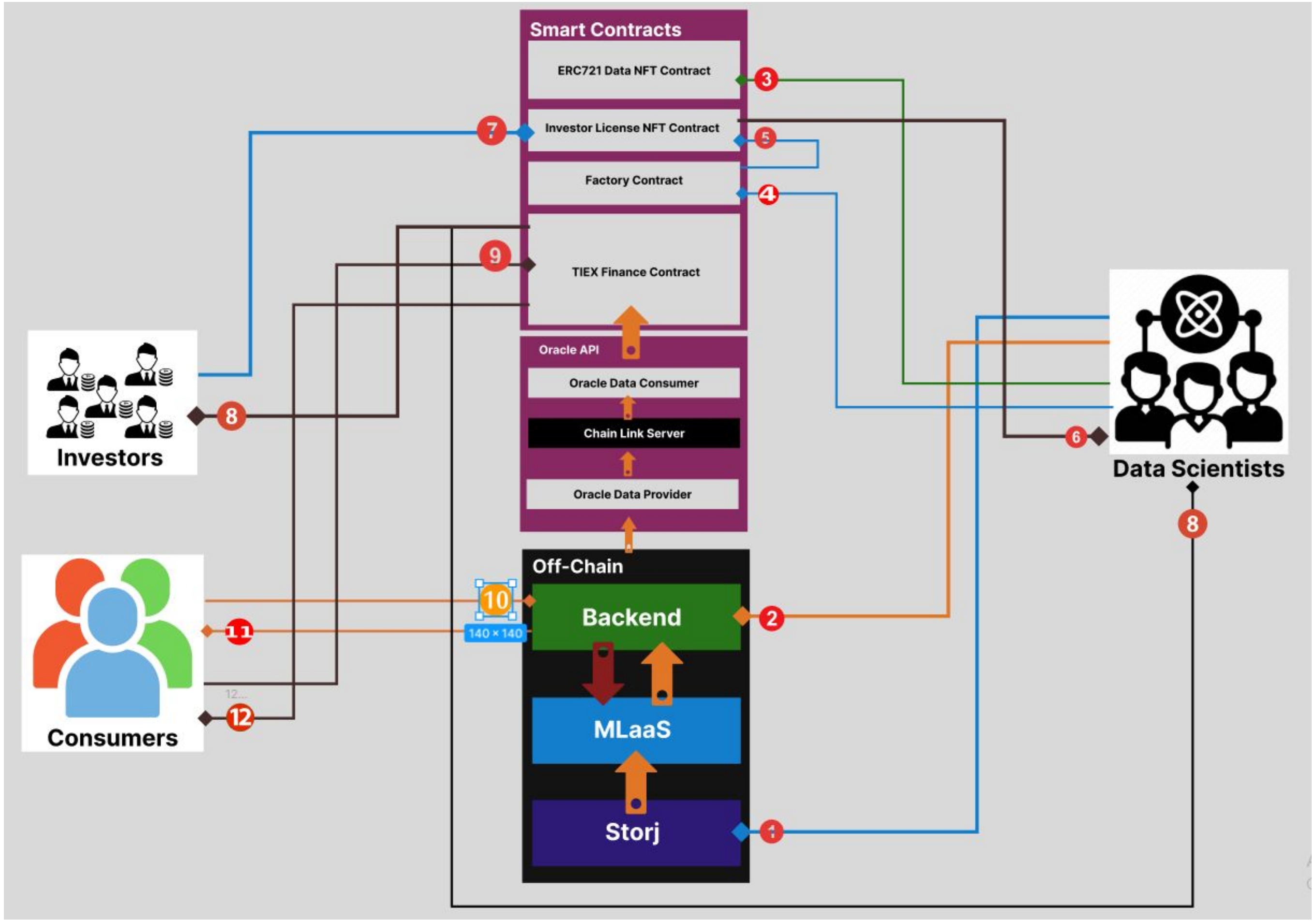


Figure: 1-2
TIEX Solution Diagram

1. Data Scientist Actions (Data Provider) [Figure 1-2 above]

① The Data Scientist directly uploads and loads Data Model and Dataset to Storj Storage through **Presigned URL**.

② The Data Scientist submits project details such as Price (TIEX) per a call, Tutorial, Social Links, Project Name, and etc.

③ Once the TIEX Admin approves the provided information and data model, datasets, the Data Scientist mints the Data NFT from the ERC721 Data NFT Contract with its own Crypto wallet such as Metamask and owns the Copyright / base IP of the Data Model by holding it.

④ If the Data Scientist wants investment from investors, he creates an ERC721 Data Investor License Contract through Factory Contract. When they create it by Factory Contract, they provide the following the parameters like **[Table 1 - 1]**.

⑤ Once Data Scientist calls the **createDataInvestorLicenseNFTContract()** function in the Factory Contract, the Factory Contract creates a new ERC721 **DataInvestorLicenseNFTContract** for the only current data model.

⑥ When the revenue of the data model reaches a certain condition, Data NFT holders can withdraw the entire invested fund from **DataInvestorLicenseNFTContract.**

2. Data Investor Actions [Figure 1-2]

⑦ The investor owns the Investor License NFT token as a share by minting the Investor License NFT token from the ERC721 Investor License NFT Contract generated by Data Scientist.

⑧ From the TIEX Finance Contract, Data Investor (Data Investor License NFT holder) and Data Scientist (Data NFT Holder) can claim TIEX tokens at any time according to their share and revenue of the data model.

3. Consumer Actions [Figure 1-2]

⑨ The Consumers (Users) deposit TIEX Tokens through the TIEX Finance Contract to access the Data Model before calling the API of Data Model.

⑩, ⑪ The consumers access the desired data model and get the necessary response using tiex.py SDK library.

⑫ The Consumers can withdraw the remaining TIEX tokens from the TIEX Finance Contract.

How does the smart contracts and other code corresponds to the logic?

Milestone 1 [in Phase 1]:

https://github.com/blockchainexpert912/next-intelligence-exchange-platform

The Intelligence Exchange Platform (TIEX Protocol)

Payment Token: TheIntelligenceExchangeToken (TIEX token)
https://testnet.bscscan.com/address/0xc9c806a2986ce19d0eb3dfbc2ceb7b8492e3f4ee#code

Data NFT Token: DataNFTToken (DNT token)
https://testnet.bscscan.com/address/0xF1E974527D845e775423CD1D6c8f6d6FD93d60D6#code

TheIntelligenceExchangeTokenContract

https://testnet.bscscan.com/address/0xc9c806A2986Ce19D0eB3Dfbc2ceb7B8492E3F4eE#code

Write Functions Comments

① approve(address spender, uint256 amount);

Sets `**amount**` as the allowance of `**spender**` over the caller's tokens

② burn(address account, uint256 amount)

Destroys `amount` tokens from `account`, reducing the total supply.

③ decreaseAllowance(address spender, uint256 subtractedValue)

Atomically decreases the allowance granted to `spender` by the caller.

This is an alternative to {approve} that can be used as a mitigation for problems described in {IERC20-approve}.

④ increaseAllowance(address spender, uint256 addedValue)

Atomically increases the allowance granted to `spender` by the caller.

This is an alternative to {approve} that can be used as a mitigation for problems described in {IERC20-approve}.

⑤ renounceOwnership()

Leaves the contract without owner. It will not be possible to call `onlyOwner` functions anymore.

Can only be called by the current owner.

NOTE: Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner.

⑥ transferOwnership(address newOwner)

Transfers ownership of the contract to a new account (`newOwner`).

Can only be called by the current owner.

⑦ transfer(address to, uint256 amount)

Moves `amount` tokens from the caller's account to `to`.

Returns a boolean value indicating whether the operation succeeded.

⑧ transferFrom(address from, address to, uint256 amount)

Moves `amount` tokens from `from` to `to` using the allowance mechanism. `amount` is then deducted from the caller's allowance.

Returns a boolean value indicating whether the operation succeeded.

Read Functions Comments

1. allowance(address owner, address spender)

Returns the remaining number of tokens that `spender` will be allowed to spend on behalf of `owner` through {transferFrom}. This is zero by default. This value changes when {approve} or {transferFrom} are called.

2. balanceOf(address account)

Returns the amount of tokens owned by `account`.

3. decimal()

Returns the decimals places of the token.

4. name()

Returns the name of the token.

5. symbol()

Returns the symbol of the token, usually a shorter version of the name.

6. owner()

Returns the address of the current owner.

7. totalSupply()

Returns the amount of tokens in existence.

DataNFTContract

https://testnet.bscscan.com/address/0xF1E974527D845e775423CD1D6c8f6d6FD93d60D6#code

Write Functions Comments

① adopt()
Mints the Data NFT and provides the Copyright / base IP of the Data Model by holding it.
② approve(address to, uint256 tokenId)
Gives permission to `to` to transfer `tokenId` token to another account. The approval is cleared when the token is transferred. Only a single account can be approved at a time, so approving the zero address clears previous approvals.
③ burn(uint256 tokenId)
Burns `tokenId`
④ pause(bool val)
Sets sale status
⑤ renounceOwnership()
Leaves the contract without owner. It will not be possible to call `onlyOwner` functions anymore. Can only be called by the current owner.

6. safeTransferFrom(address from, address to, uint256 tokenId, bytes calldata data)

Safely transfers `tokenId` token from `from` to `to`.

7. setApprovalForAll(address operator, bool approved)

Approves or removes `operator` as an operator for the caller.

Operators can call {transferFrom} or {safeTransferFrom} for any token owned by the caller.

8. setBaseURI(string memory baseURI)

sets the base URI for all token IDs. It is automatically added as a prefix to the value returned in tokenURI, or to the token ID if tokenURI is empty.

9. setMintPrice(uint256 _newPrice)

Sets the cost for minting Data NFT

10. setPaymentToken(address _newPaymentToken)

Sets the payment token

11. transferFrom(address from, address to, uint256 tokenId)

Transfers `tokenId` token from `from` to `to`.

12. transferOwnership(address newOwner)

Transfers ownership of the contract to a new account (`newOwner`).

Can only be called by the current owner.

13. withdraw(uint256 _amount, IERC20 _token)

Withdraws TIEX tokens

Read Functions Comments

1. balanceOf(address owner)

Returns the number of tokens in ``owner``'s account.

2. getApproved(uint256 tokenId)

Returns the account approved for `tokenId` token.

3. getMintPrice()

Returns current Data NFT mint price

4. getPause()

Returns pause status

5. getPaymentToken

Returns payment token address

6. isApprovedForAll(address owner, address operator)

Returns if the `operator` is allowed to manage all of the assets of `owner`.

7. name()

Returns the name of the token.

8. owner()

Returns the address of the current owner of Data NFT Contract

9. ownerOf(uint256 tokenId)

Returns the owner of the NFT specified by tokenId.

10. symbol()

Returns the token collection symbol.

11. tokenByIndex(uint256 index)

Returns a token ID at a given `index` of all the tokens stored by the contract.

12. tokenOfOwnerByIndex(address owner, uint256 index)

Returns a token ID owned by `owner` at a given `index` of its token list.

13.tokenURI(uint256 tokenId)

Returns the Uniform Resource Identifier (URI) for tokenId token.

14. totalSupply()

Returns the total amount of tokens stored by the contract.

15. walletOfOwner(address owner)

Returns the NFT tokenIds that the owner has

How the functionality and the smart contract can be deployed, scaled and maintained

1. Development Environment & Deployment

Language: Solidity

Framework: Hardhat (plus Remix)

Unit-test: Chai Framework

Network: EVM Networks

Deployment: Hardhat (plus Remix)

Environment: Node.JS 18.x, Hardhat, VS Code, Solidity Extensions

2. Improvement & Maintenance

Smart contracts on Ethereum are self-executing programs that run in the Ethereum Virtual Machine (EVM). These programs are immutable by design, which prevents any updates to the business logic once the contract is deployed.

How do I upgrade a Contract?

The basic idea is to use a proxy to make upgrades. The first contract is a simple wrapper or “proxy” which users interact with directly and is in charge of forwarding transactions to and from the second contract, which contains the logic.

Using ERC-1167: Minimal Proxy Contract, we can build upgradable Smart Contract.

Thanks for your review

We want feedback from you …

1st March 2023