

Artwork Authenticity Verification using Hyperledger Fabric

Name-Vibhushit Tyagi

Email- vibhushit.tyagi@npci.org.in

Table of Contents

- About The Project**
 - Problem
 - Solution
- Workflow**
- Application**
 - Prerequisites
 - How to run the Application
- Chaincode Functions**
- Resources and technologies Used**

About The Project

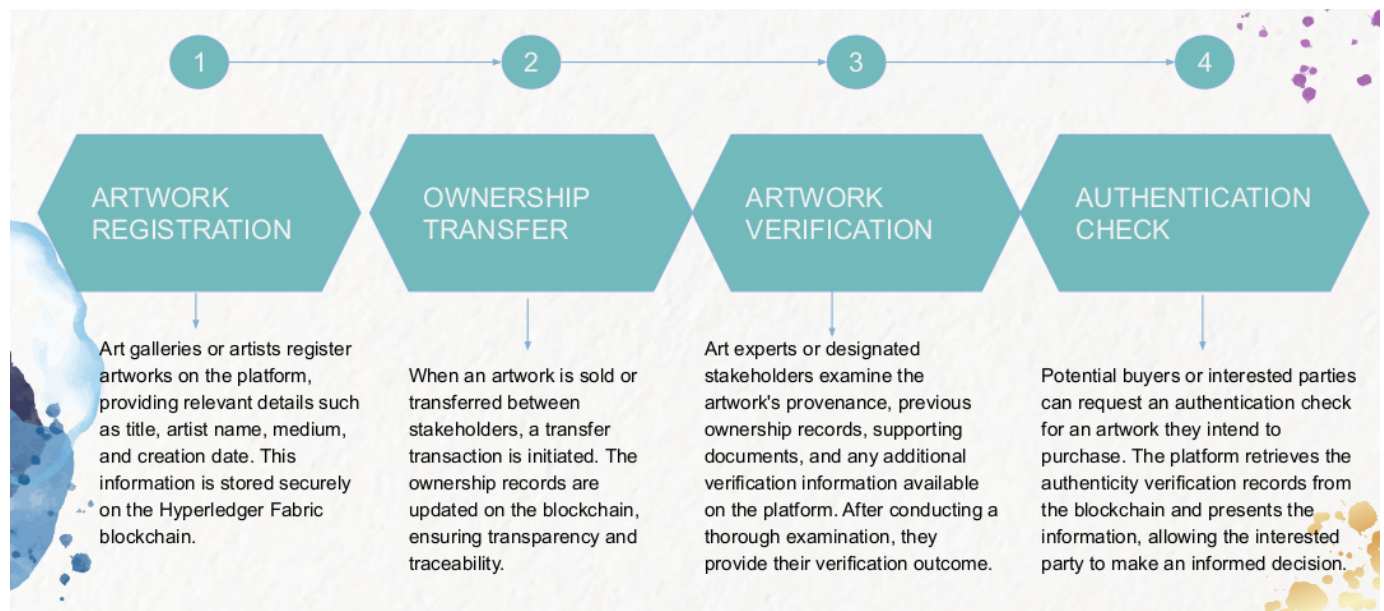
Problem

The art market faces significant challenges when it comes to verifying the authenticity and provenance of artworks. The current centralized systems lack transparency, making it difficult to trace an artwork's history and validate its authenticity. Additionally, fraudulent practices, such as art forgery and unauthorized ownership transfers, undermine the trust of collectors, galleries, and buyers.

Solution

Hyperledger Fabric is chosen as the underlying blockchain framework for the art authenticity verification system due to its permissioned network structure, support for smart contracts, privacy features, and scalability. Fabric ensures that only authorized participants can access and validate the authenticity information, while maintaining transparency and immutability.

Workflow



1. Organization 1: Art Expert

- The Art Expert organization is responsible for verifying the authenticity of paintings.
- It can create, read, update, and delete expert records using the ExpertContract.
- It can also perform the verifyAuthenticity transaction on the PaintingContract to verify the authenticity of a painting.

2. Organization 2: Artist

- The Artist organization represents the artists who create paintings.
- It can create, read, update, and delete painting records using the PaintingContract.
- Artists can register their paintings by calling the createPainting transaction, providing details such as the artist's name, title of the painting, and the year it was created.

3. Organization 3: Art Gallery

- The Art Gallery organization manages the collection of paintings.
- It can create, read, update, and delete art gallery records using the ArtGalleryContract.
- Art galleries can register paintings in their collection by calling the registerPainting transaction, providing the painting ID.

The overall flow would typically involve the following steps:

1. Art Expert verifies the authenticity of a painting:

- The Art Expert performs the verifyAuthenticity transaction on the PaintingContract to verify the authenticity of a painting.
- The transaction updates the verified flag of the painting record to indicate its authenticity.

2. Artists create and manage paintings:
 - Artists from the Artist organization can create paintings by invoking the createPainting transaction on the PaintingContract.
 - They can also read, update, and delete their painting records using the respective functions in the PaintingContract.
3. Art Galleries register and manage paintings:
 - Art Galleries from the Art Gallery organization can register paintings in their collection by calling the registerPainting transaction on the ArtGalleryContract.
 - They can also read, update, and delete art gallery records using the respective functions in the ArtGalleryContract.

Application

Prerequisites

Introduction to Blockchain with Industry Applications Blockchain Foundation Program

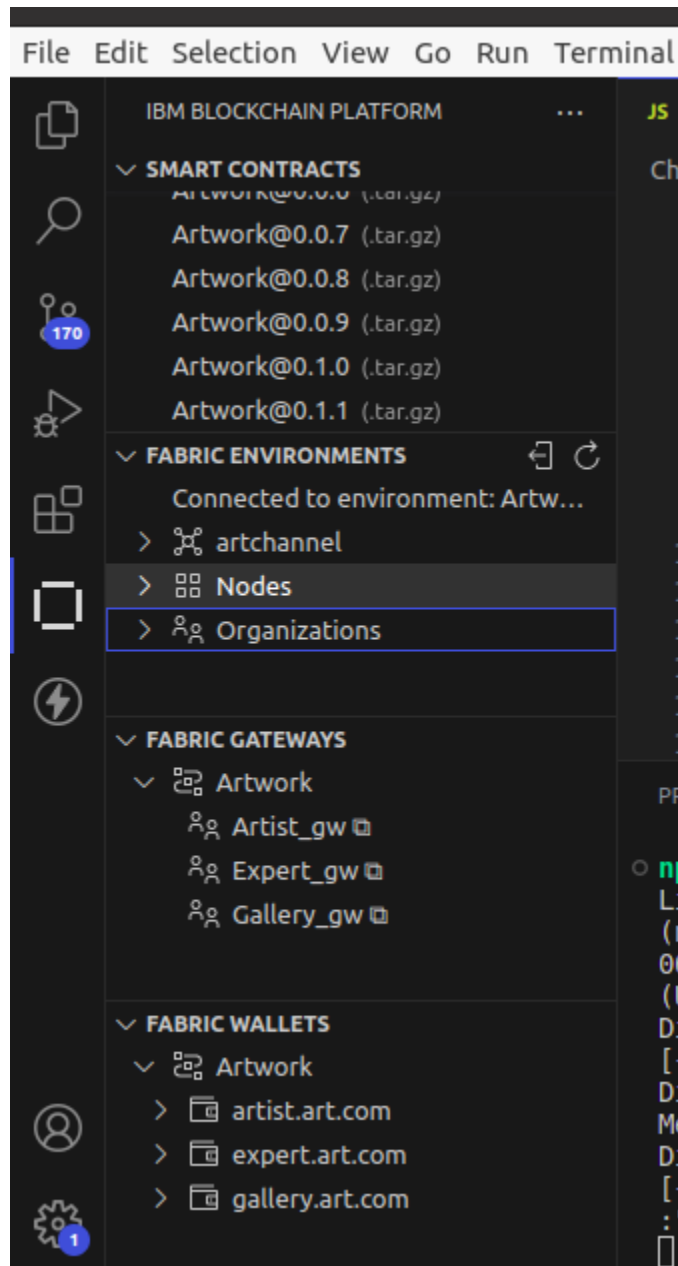
How to run the Application

1. Open the terminal and navigate to the "Network" folder.
2. Run the following command to install all the required dependencies:
(./installDependencies.sh)
3. Next, set up the "bin" folder by running the following command:
(./installDependencies.sh bin) , This step will configure the "bin" folder, which contains various binaries required for network operations.
4. Start the Hyperledger Fabric network by running the following command: (./startNetwork.sh) , This command will initiate the minifab network, allowing you to interact with the network and deploy smart contracts.
5. Launch Visual Studio Code and manually add the IBM Extension. If you don't have the IBM Extension, you can download it from this link:
https://gitlab.com/CHF_KBA/kba_chf_ibmblockchainplatformextension_vscode/-/raw/main/ibm-blockchain-platform-2.0.8.vsix?inline=false.
6. Configure the wallets, environment, and gateways in the IBM Extension. This will establish the necessary connections and configurations to interact with the network.
7. Open the "Artwork" folder in Visual Studio Code, located in the chaincode directory. This folder contains the smart contract code for the Artwork contract.
8. Use the IBM Extension to package the smart contract. This step bundles the contract code and dependencies into a deployable package.
9. Deploy the smart contract onto the network using the IBM Extension. This will instantiate the contract on the network and make it available for transaction execution.
10. Utilize the configured gateways in the IBM Extension to perform various transactions on the network. You can execute functions defined in the smart contract and interact with the deployed contract.

By following these steps, you will have successfully set up the Hyperledger Fabric network, deployed the smart contract, and be ready to perform transactions using the gateways.

Then Final Project Output:

IBM Extension Connections Screenshot:



Create Painting Transaction from Painting Contract:

SMART CONTRACTS

Artwork@0.0.0 (.tar.gz)

Artwork@0.0.7 (.tar.gz)

Artwork@0.0.8 (.tar.gz)

Artwork@0.0.9 (.tar.gz)

Artwork@0.1.0 (.tar.gz)

Artwork@0.1.1 (.tar.gz)

FABRIC ENVIRONMENTS

Connected to environment: Artw...

> artchannel

> Nodes

> Organizations

FABRIC GATEWAYS

paintingExists

createPainting

readPainting

updatePainting

deletePainting

> ExpertContract

FABRIC WALLETS

> Artwork

> artist.art.com

Create transaction

Contract selection

PaintingContract

Manual inputTransaction data dir...

Transaction name

createPainting

Transaction arguments

```
{  "arg0": "P02",  "arg1": "Pablo",  "arg2": "Mexican Dreams",  "arg3": "2023"}
```

Read Painting Transaction from Painting Contract:

Contract selection

PaintingContract

Manual inputTransaction data dir...

Transaction name

readPainting

Transaction arguments

```
{  "arg0": "P02"}
```

Transaction output

Returned value from readPainting: {"artist":"Pablo","title":"Mexican Dreams","verified":false,"yearCreated":"2023"}

VerifyAuthenticity Transaction from ExpertContract:

Contract selection

ExpertContract

Manual inputTransaction data dir...

Transaction name

verifyAuthenticity

Transaction arguments

```
{
  "arg0": "P02"
}
```

Transaction output

No value returned from verifyAuthenticity

QueryAllPaintings Transaction from GalleryContract:

Contract selection

ArtGalleryContract

Manual inputTransaction data dir...

Transaction name

queryAllPaintings

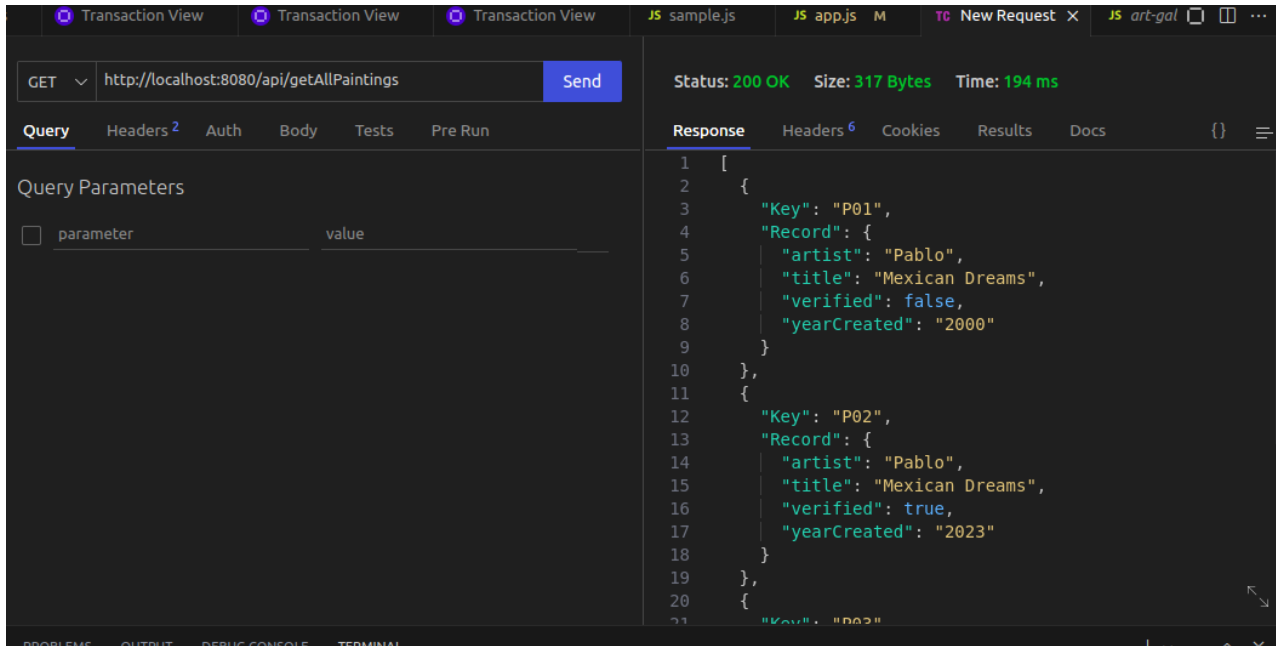
Transaction arguments

```
[]
```

Transaction output

Returned value from queryAllPaintings: [{"Key":"P01","Record":{"artist":"Pablo","title":"Mexican Dreams","verified":false,"yearCreated":"2000"}},{ "Key":"P02","Record":{"artist":"Pablo","title":"Mexican Dreams","verified":true,"yearCreated":"2023"}},{ "Key":"P03","Record":{"artist":"Pablo","title":" Mexican Nights ","verified":false,"yearCreated":"2023"}}]

API call to read all paintings data:



Transaction View Transaction View Transaction View JS sample.js JS app.js M TC New Request X JS art-gal

GET

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

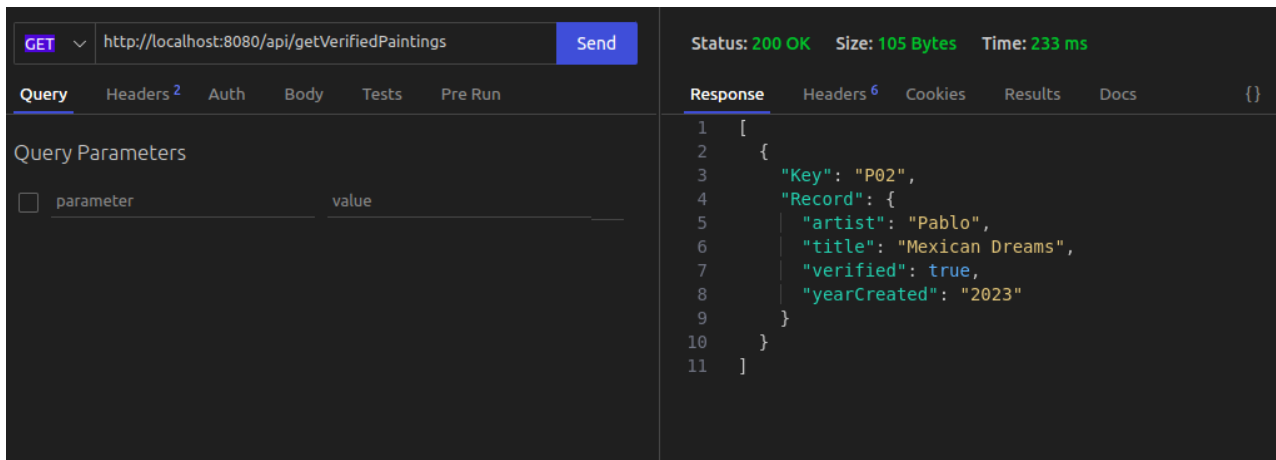
parameter	value
-----------	-------

Status: 200 OK Size: 317 Bytes Time: 194 ms

Response Headers 6 Cookies Results Docs {}

```
1 [
2   {
3     "Key": "P01",
4     "Record": {
5       "artist": "Pablo",
6       "title": "Mexican Dreams",
7       "verified": false,
8       "yearCreated": "2000"
9     }
10  },
11  {
12    "Key": "P02",
13    "Record": {
14      "artist": "Pablo",
15      "title": "Mexican Dreams",
16      "verified": true,
17      "yearCreated": "2023"
18    }
19  },
20  {
21    "Key": "P03"
```

API call to get all verified paintings data:



GET

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

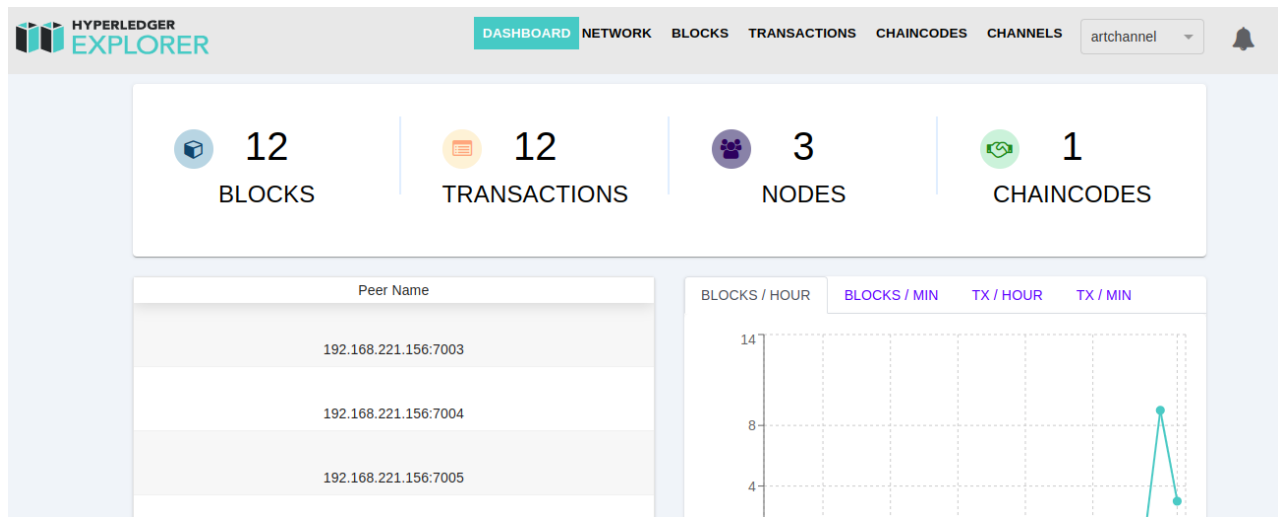
parameter	value
-----------	-------

Status: 200 OK Size: 105 Bytes Time: 233 ms

Response Headers 6 Cookies Results Docs {}

```
1 [
2   {
3     "Key": "P02",
4     "Record": {
5       "artist": "Pablo",
6       "title": "Mexican Dreams",
7       "verified": true,
8       "yearCreated": "2023"
9     }
10  }
11 ]
```

Blockchain using minifab explorer:



Couch DB values:

id	key	value
<input type="checkbox"/> P01	P01	{ "rev": "1-b43fb847fd1ee24a8817de8..." }
<input type="checkbox"/> P02	P02	{ "rev": "2-5b110503cc4a02d650c8f17..." }
<input type="checkbox"/> P03	P03	{ "rev": "1-36ec3db5a9069a279e3a7a..." }

```
1 {
2   "_id": "P01",
3   "_rev": "1-b43fb847fd1ee24a8817de858fa67c4e",
4   "artist": "Pablo",
5   "title": "Mexican Dreams",
6   "verified": false,
7   "yearCreated": "2000",
8   "~version": "CgMBCAA="
9 }
```

Chaincode Functions

ArtGalleryContract:

1. queryAllPaintings: Retrieves all paintings stored in the ledger.
2. queryVerifiedPaintings: Retrieves only the verified paintings stored in the ledger.

PaintingContract:

1. createPainting: Creates a new painting with the provided details.
2. readPainting: Retrieves the details of a specific painting using its ID.
3. updatePainting: Updates the value of a painting with a new value.
4. deletePainting: Deletes a painting from the ledger.

ExpertContract:

1. verifyAuthenticity: Performs authenticity verification for a specific painting.
2. getPaintingAuthenticity: Retrieves the authenticity verification status of a painting.

RESOURCES USED:

VS code, IBM extension for blockchain, Internet, Minifab(Hyperledger Fabric).