Ira A. Fulton Schools of Engineering
Arizona State University

CSE 598: Engineering Blockchain Applications

Creating an ERC-721 Token Standard Smart Contract on Ethereum Project

Purpose

This project will help you familiarize yourself with the Ethereum (EVM- Ethereum Virtual Machine) ecosystem by writing, testing, and deploying an ERC-721 Token Standard smart contract on the Polygon Mumbai Testnet in the Solidity programming language. Through this project, you will learn foundational knowledge to begin your journey of creating EVM-based Web3 labs that utilize token mechanisms including Decentralized Autonomous Organizations (DAOs) and Non-Fungible Tokens (NFTs).

Objectives

Learners will be able to:

- Develop an EVM-based Web3 application
- Deploy and ERC-721 Token Standard smart contracts an Ethereum-based network using Solidity

Technology Requirements

- Solidity
- Remix IDE
- Browser (Chrome, Firefox, Brave, and Edge)

Project Description

We will be deploying the code on Polygon Mumbai Testnet which is a part of EVM Ecosystem. All the development is similar to Ethereum-based environments but only the deployment part changes as you will be deploying in Polygon Testnet. The currency used here is MATIC instead of Ether.
Tokens are widely used in the Ethereum ecosystem to represent many things including governance power, in-game currencies, reputation points, and much more. The Ethereum ecosystem uses Ethereum Improvement Proposals (EIPs) for the community to propose standards specifying potential

new features or processes for the network.

EIPs play a central role in how changes happen and are documented on Ethereum. They are the way for people to propose, debate, and adopt changes. There are different types of EIPs including core EIPs for low-level protocol changes that affect consensus and require a network upgrade as well as Ethereum Request for Comments (ERCs) for application standards. One such type of ERC standard is the token standard, like the ERC-721 Token Standard, which allows applications interacting with the specific ERC standard token to treat all other tokens using the same rules, which makes it easier to create interoperable applications.

The ERC standard this lab will use is the ERC-721 Token Standard, the most widely used non-fungible token standard currently on the Ethereum network. Fungibility refers to the ability for each token to be treated exactly the same, in contrast to non-fungible tokens in which the token has properties that make it distinct from other sets of tokens.

A smart contract on the Ethereum Network can be considered an ERC-721 Token Contract if it implements the following methods and events (for further reading on the standard, visit https://ethereum.org/en/developers/docs/standards/tokens/ERC-721/).

The rest of this lab documentation will walk you through the steps you will need to take to get acquainted with the tools and resources that will allow you to write, test, and deploy your own ERC-721 Token Standard Smart Contract on the Polygon Mumbai Testnet.

The tasks below are separated into two categories: Set-Up Tasks and Graded Tasks. Set-Up Tasks guide you to the resources, tools, and documentation that will help you complete the lab. Graded Tasks define how the smart contract you submit is expected to function for the system to grade. The tools and resources recommended in the Set-Up tasks will allow you to test that your smart contract is working properly so that you know what grade to expect upon submission (please ask the course team questions along the way, we want all smart contracts submitted to be properly functioning ERC-721 Token Standard Smart Contracts that you can be proud of)! Happy web3 development!

Directions

Setup Tasks

Task 1 - MetaMask

To deploy a smart contract on a Polygon Mumbai Network, you will first need a way to connect to a Polygon Mumbai Network and pay the gas fees associated with deploying your smart contract (it costs the network computational power and storage to execute transitions, those engaging in

transactions pay for these costs using “gas”).

It is recommended to use the MetaMask browser extension (supported browsers include Chrome, Firefox, Brave, and Edge), as it gives us an easy-to-use interface into Ethereum Networks that can be used for this lab, as well as providing you a wallet and wide-ranging interoperability with other smart contracts and ecosystems you may be interested in exploring in the future.

You can download the browser extension by going to this link: MetaMask Download

Once you’ve downloaded the browser extension and gone through the setup process, you are ready for the next task.

Task 2 - Polygon Mumbai Testnet

You will be deploying your smart contracts to the Polygon Mumbai Testnet. The Polygon Mumbai Testnet allows you to deploy your smart contract in a live setting, without the need for real currency and other mainnet security considerations as you are engaging in development. Testnets are widely used and it is highly recommended to test and deploy smart contracts on a testnet before you deploy on the mainnet.

You can learn more about Ethereum Networks by going to this link (the Polygon network will be used): https://ethereum.org/en/developers/docs/networks/

And Polygon Network here - https://wiki.polygon.technology/docs/integrate/network/

To set up the Polygon Mumbai Testnet on MetaMask, open this link - ChainList.

You can add Mumbai quickly without having to enter details manually. Then follow the pop ups to add the network. On the main pop-up of Metamask , if you click the Ethereum Mainnet option you can now switch to the Polygon Mumbai Test Network. Once switched, you have now completed the steps necessary in this task to be able to connect to the Polygon Mumbai Test Network via your browser.

Task 3 - Polygon Mumbai Testnet Faucet

The on-chain currency for the Polygon Mumbai Testnet is MATIC. You will need MATIC to pay gas fees when deploying your smart contract and transacting on the Polygon Mumbai network.

1. Navigate to the Faucet: Go to the LearnWeb3 Mumbai Faucet page.

2. Sign In to LearnWeb3: Click on the "Sign In" button located in the top-right corner of the page.

3. Connect Your GitHub/Other SM Account: If you haven't already connected your GitHub account to LearnWeb3, you will need to do so to gain access to the faucet.

4. Enter Your Wallet Address: Once signed in and your GitHub is connected, enter your MetaMask wallet address into the appropriate field on the faucet page.

5. Request MATIC: Click the button to request MATIC tokens. The faucet will then process your request and send the MATIC to your wallet address.

6. Receive MATIC: The MATIC should be sent to your wallet shortly after your request. This can take a few minutes to several hours, depending on the faucet's processing time and current demand.

7. Check Your Balance: Once the transaction is complete, you can check your MetaMask wallet to confirm that you have received the MATIC.

Task 4 - Remix IDE and Solidity

It is recommended to use the Remix IDE for this lab (https://remix.ethereum.org). Remix is a browser-based IDE that easily connects to MetaMask, allowing you to develop, test, and deploy your Solidity smart contract, all directly in your browser.

The best way to get familiar with any new development platform is to explore and try it out for yourself! Remix provides you with example contracts for you to explore in the default workspace (found on the left in your File Explorers tab) and detailed documentation here: https://remix-ide.readthedocs.io/en/latest/#

We will be helping you navigate and use Remix through our Solidity development live sessions. If you would like to explore Solidity development on your own, the Solidity documentation is well-detailed and can be found here: https://docs.soliditylang.org/en/v0.8.12/

Setup demo video: Metmask Setup

Coursework

Part 1

This part will **not be graded but is required** in order to complete the rest of the project.

Before getting started on the task, you will require a set of parameters (Token name, symbol, and message) that you are expected to use so that each method returns to the desired output. For example, you will be given the strings that your smart contract should return when the name() method is called.

To get your token strings,

1. In the course, go to the Canvas Assignment, "**Submission: Part 1 - Creating an ERC-721 Token Standard Smart Contract on Ethereum Project)**"
2. Click the "**Load Submission..in new window**" button
3. Once in Gradescope, select the project titled "**Part 1 - Creating an ERC-721 Token Standard Smart Contract on Ethereum Project**" and a pop-up window will appear.
4. In the pop-up:
    a. Submit a "**tokens.txt**" which contains
        i. Your **Full Name** as per in the course
        ii. Your **ASURITE username** (e.g., sparky123)
        iii. Format your content as: **Name,asurite**
    b. Click "**Upload**" to get your tokens.

Part 2

This part of the project will be auto-graded. The following graded tasks define what the expected output should be when you submit your deployed ERC-721 smart contract address and the methods/events are tested:

- **Graded Task 1 - Token Name**: When called, the contract should return the string you received from Part 1 for the name value in Task 1.
- **Graded Task 2 - Token Symbol**: This returns the symbol used to denote your token. When called, the contract should return the string you received from Part 1 for the token symbol in Task 2.
- **Graded Task 3 - Message**: This returns the message used to denote your token. When called, the contract should return the string you received from Part 1 for the token message in Task 3. "**message**" is the key name to use.

Please deploy this ERC-721 Token Standard Smart Contract to the Mumbai Testnet via Remix and Metamask. After deployment, save the deployed smart contract address for submission.

Part 3:

Submit a file of your code showing the work you developed.

Submission Directions for Project Deliverables

The autograder should not be used to debug or test function-by-function as you develop your ERC-721 token. All testing of ERC-721 functionality can happen through remix and does not require deployments to the Polygon Mumbai test network (which costs gas). The autograder should be used when you believe you have a working ERC-721 contract and have gone through all the necessary steps to be graded.

Preparing the Deliverables

You will prepare and submit two (2) files for this project.

Part 2 Submission

You must submit one (1) text file named "**submission.txt**" for this part which contains:

- Your **ASURite username** (e.g., sparky123)
- **Smart Contract Address**
- The format of the two items must be "**asurite_username,smart_contract_address**"

Part 3 Submission

You will submit one (1) file as a **PDF** for this part which contains:

- **Program/Code**: Please include the code you have written for the project.
    - Please include your name, course title, and date with your writing within the file
    - Your file should be titled using the format: "Last Name_First Name_CSE598 EBA_Creating an ERC-721 Project"

Submitting to Gradescope

Your submissions will be graded automatically. You must submit your final work to Gradescope to receive credit. You will have unlimited submission attempts, but the course team will only review the latest submission.

1. In the course, co to the Canvas Assignment, "**Submission: Part 2 & 3 - Creating an ERC-721 Token Standard Smart Contract on Ethereum Project**".
2. Click the "**Load Submission…in new window**" button.

3. Once in Gradescope, select the project titled "**Part 2 & 3 - Creating an ERC-721 Token Standard Smart Contract on Ethereum Project**" and a pop-up window will appear.
4. In the pop-up window, submit your "**submission.txt**" and **PDF code** file together with the contents mentioned in the deliverables section for the grading.
5. If needed: to resubmit the project in Gradescope:
    a. Return to the Canvas submission and open Gradescope.
    b. You will be navigated to the "Autograder Results" page (if it is not your first submission).
    c. Click the "**Resubmit**" button on the bottom right corner of the page and repeat the process from Step 3.

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Gradescope into your grade.

Evaluation

We will fetch the NFT information by interacting with the smart contract address provided by you and parse through the key values and verify if the tokens obtained match the tokens we provided in Part 1.

This project is worth a total of **210 points**. The autograder will directly call the methods and events on your live contract that is deployed on the testnet as specified in the instructions above. The output will be your grade. It will also show which methods/events passed or failed to help you debug.

- 50 points: If the token name is correct
- 50 points: If the symbol is correct
- 60 points: If the message is correct
- 50 points: Code file is submitted as a PDF and accurate to your work