**DSCTransferHub API Documentation (evm\_dsc\_transfer\_hub.py)**

**Overview**

The evm\_dsc\_transfer\_hub.py script is a FastAPI application that provides a RESTful API for interacting with the DSCTransferHub smart contract on the Ethereum Sepolia testnet (chain ID 11155111). The contract enables cross-chain token transfers (e.g., DSC) to Avalanche Fuji (chain selector 14767482510784806043) using Chainlink CCIP. It supports depositing tokens and ETH, transferring tokens cross-chain, and withdrawing token deposits. The API is designed for secure blockchain interactions and uses environment variables for configuration.

This document is intended for:

* **Frontend Engineers**: To integrate the API into user interfaces, understand endpoint inputs, outputs, and error handling.
* **AI Engineers**: To understand contract interactions, extend cross-chain functionality, and debug issues like transaction reverts or library errors.

**Setup**

**Prerequisites**

* **Python**: Version 3.8 or higher.
* **Dependencies**: Requires fastapi, uvicorn, web3, python-dotenv, and pydantic. Ensure web3.py is version 6.0.0 or higher to avoid errors like missing attributes.
* **Sepolia Node**: Access to a Sepolia RPC endpoint (e.g., Infura, Alchemy).
* **Environment Variables**: Create a .env file in the project root with:
  + SEPOLIA\_RPC\_URL: Sepolia node endpoint (e.g., https://sepolia.infura.io/v3/YOUR\_PROJECT\_ID).
  + PRIVATE\_KEY: Ethereum private key (with 0x prefix) for signing transactions.
  + EVM\_DSC\_TRANSFER\_HUB\_ADDRESS: DSCTransferHub contract address on Sepolia.

**Running the API**

1. Save the script as evm\_dsc\_transfer\_hub.py in your project directory (e.g., /mnt/d/chainlink-CrossChain-Stablecoin/Base-Hackathon-Stable-Token/dsc-api).
2. Start the FastAPI server using the command: uvicorn evm\_dsc\_transfer\_hub:app --host 0.0.0.0 --port 8000 --reload.
3. Access the API at http://localhost:8000. Use /docs for the interactive Swagger UI to test endpoints.

**Avoiding Common Errors**

* **Missing Attributes**: Ensure no local file named web3.py or Web3.py exists in the project directory, as it can conflict with web3.py. Verify web3.py version is 6.0.0 or higher.
* **Contract Not Deployed**: Confirm the DSCTransferHub contract is deployed at the specified address using a blockchain explorer like Etherscan.
* **Nonce Issues**: The API retries transactions up to three times if a "nonce too low" error occurs, common in Sepolia due to network congestion.
* **Network Connection**: Ensure the Sepolia RPC endpoint is accessible and the chain ID is 11155111.

**Endpoints**

**1. Approve Tokens (/approve-tokens)**

* **Method**: POST
* **Description**: Approves the DSCTransferHub contract to spend tokens (e.g., DSC) on behalf of the user’s wallet. This is a prerequisite for depositing tokens for cross-chain transfers.
* **Request Body**:
  + token\_address: The address of the token to approve (e.g., DSC contract address).
  + amount: The amount of tokens to approve (in token units, e.g., 1.0 for 1 DSC).
  + Example:
  + {
  + "token\_address": "0x...DSC\_ADDRESS...",
  + "amount": 1.0
  + }
* **Response**:
  + Success (approval needed): Returns the transaction hash of the approval.
  + {"approval\_tx\_hash": "0x..."}
  + Success (no approval needed): Indicates sufficient allowance exists.
  + {"message": "No approval needed; sufficient allowance already set"}
  + Error: Details the issue (e.g., invalid address or unsupported token).
  + {"detail": "Token not supported by DSCTransferHub"}
* **Status Codes**:
  + 200: Approval succeeded or was not needed.
  + 400: Invalid token address, non-positive amount, or token not supported by DSCTransferHub.
  + 500: Transaction failure or server error (e.g., network issue).
* **Notes**:
  + Checks if the token is supported by DSCTransferHub before approving.
  + Verifies existing allowance to avoid redundant transactions.
  + Must be called before /deposit-tokens or /transfer-tokens.

**2. Deposit Tokens (/deposit-tokens)**

* **Method**: POST
* **Description**: Deposits tokens (e.g., DSC) into the DSCTransferHub contract for cross-chain transfer preparation. Tokens must be approved first.
* **Request Body**:
  + token\_address: The address of the token to deposit (e.g., DSC contract address).
  + amount: The amount of tokens to deposit (in token units, e.g., 0.5 for 0.5 DSC).
  + Example:
  + {
  + "token\_address": "0x...DSC\_ADDRESS...",
  + "amount": 0.5
  + }
* **Response**:
  + Success: Returns the transaction hash of the deposit.
  + {"tx\_hash": "0x..."}
  + Error: Details the issue (e.g., insufficient allowance).
  + {"detail": "Insufficient allowance. Approve tokens first"}
* **Status Codes**:
  + 200: Deposit succeeded.
  + 400: Invalid token address, non-positive amount, insufficient allowance, or unsupported token.
  + 500: Transaction revert (e.g., contract logic error) or server error.
* **Notes**:
  + Requires prior approval via /approve-tokens.
  + Verifies token support using DSCTransferHub’s isSupportedToken function.
  + Deposits are tracked in userTokenDeposits for the user and token.

**3. Send ETH (/send-eth)**

* **Method**: POST
* **Description**: Sends ETH to the DSCTransferHub contract to deposit funds for cross-chain transfer fees, triggering the contract’s receive() function.
* **Request Body**:
  + amount: The amount of ETH to send (in ETH units, e.g., 0.1 for 0.1 ETH).
  + Example:
  + {
  + "amount": 0.1
  + }
* **Response**:
  + Success: Returns transaction details, including balances and events.
  + {
  + "tx\_hash": "0x...",
  + "eth\_balance\_before": 1.5,
  + "eth\_balance\_after": 1.39,
  + "eth\_deposits": 0.1,
  + "events": {
  + "deposit\_events": [{"from": "0x...", "token": "0x...", "amount": 0.1}]
  + },
  + "tx\_data": "0x"
  + }
  + Error: Details the issue (e.g., insufficient ETH).
  + {"detail": "Insufficient ETH balance: 1.0 ETH available, 1.1 ETH required"}
* **Status Codes**:
  + 200: ETH deposit succeeded.
  + 400: Non-positive amount or insufficient ETH balance (including gas).
  + 500: Transaction revert or server error.
* **Notes**:
  + Checks ETH balance before sending, accounting for gas costs.
  + Parses TokensDeposited events to confirm deposit.
  + Uses a gas limit of 100,000 and doubled gas price for reliability.
  + ETH deposits are tracked in userEthDeposits.

**4. Transfer Tokens (/transfer-tokens)**

* **Method**: POST
* **Description**: Transfers tokens cross-chain to a receiver on Avalanche Fuji (chain selector 14767482510784806043) using Chainlink CCIP. Requires sufficient token and ETH deposits.
* **Request Body**:
  + receiver: The recipient’s address on Avalanche Fuji.
  + token\_address: The address of the token to transfer (e.g., DSC).
  + amount: The amount of tokens to transfer (in token units, e.g., 0.5).
  + Example:
  + {
  + "receiver": "0x...FUJI\_ADDRESS...",
  + "token\_address": "0x...DSC\_ADDRESS...",
  + "amount": 0.5
  + }
* **Response**:
  + Success: Returns the transaction hash of the transfer.
  + {"tx\_hash": "0x..."}
  + Error: Details the issue (e.g., insufficient deposits).
  + {"detail": "Insufficient token deposit"}
* **Status Codes**:
  + 200: Transfer succeeded.
  + 400: Invalid receiver or token address, non-positive amount, unsupported token, unallowlisted chain, or insufficient token/ETH deposits.
  + 500: Transaction revert or server error.
* **Notes**:
  + Validates token support and whether Avalanche Fuji is allowlisted.
  + Checks userTokenDeposits and userEthDeposits for sufficient funds.
  + Triggers TokensTransferred event, which includes the CCIP message ID.

**5. Withdraw Deposits (/withdraw-deposits)**

* **Method**: POST
* **Description**: Withdraws token deposits from DSCTransferHub to the user’s wallet. ETH withdrawals are not supported by this endpoint.
* **Request Body**:
  + token\_address: The address of the token to withdraw (e.g., DSC). Optional, but currently required for token withdrawals.
  + Example:
  + {
  + "token\_address": "0x...DSC\_ADDRESS..."
  + }
* **Response**:
  + Success: Returns the transaction hash of the withdrawal.
  + {"tx\_hash": "0x..."}
  + Error: Details the issue (e.g., no deposits).
  + {"detail": "Nothing to withdraw"}
* **Status Codes**:
  + 200: Withdrawal succeeded.
  + 400: Invalid token address, unsupported token, or no deposits available.
  + 500: Transaction revert or server error.
* **Notes**:
  + Checks userTokenDeposits to confirm available balance.
  + Does not support ETH withdrawals in this implementation.

**Verify Transaction at Chainlink Explorer**