ðªµ Console.log

Console.log in Fhenix's Localfhenix Environmentâ

In Fhenix's Localfhenix environment, theConsole.log function and its variants serve as essential debugging tools for Solidity developers. These logs are directed to the Docker log output, aiding in monitoring and troubleshooting smart contract execution in real-time.

Public Functionsâ

TheConsole library provides two primary public logging functions:

1. log(int256 p0)
2. : Logs an integer value.
3. logBytes(bytes memory p0)
4. : Logs a byte array.

Usage Examplesâ

Here is how you can use these logging functions in your smart contracts:

Logging an Integer:

import

{ Console }

from

"@fhenixprotocol/contracts/utils/debug/Console.sol" ;

contract

ExampleContract

{ function

logIntExample ( )

public

pure

{ Console . log (123) ;

// Contract Log: 123 } } Logging a Byte Array:

import

{ Console }

from

"@fhenixprotocol/contracts/utils/debug/Console.sol" ;

contract

ExampleContract

{ function

logBytesExample ( )

public

pure

{ bytes

memory data =

"Hello, Fhenix!" ; Console . logBytes (data) ;

// Contract Log: Hello, Fhenix! } }

Usefulness in Encrypted Number Handlingâ

When working with encrypted numbers in smart contracts, having robust logging mechanisms is indispensable. Encrypted computations can be complex and opaque, making it difficult to trace issues or verify the correctness of computations. Here's how the logging functions provided by theConsole library can be particularly useful:

1. Transparency and Debugging:
2. Encrypted numbers typically undergo various transformations and operations. Logging these values at different stages helps verify that transformations are accurate and that no data corruption occurs. For instance, if an encrypted number is not decrypting correctly, logs can help trace back to the point where an issue might have arisen.
3. Validation:
4. Smart contracts that operate with encryption often involve sensitive data and critical operations. Logging intermediate values ensures that all operations are performed correctly, and their outcomes match expected results, providing an additional layer of validation.

Here's an example demonstrating how logging might be used in the context of encrypted number operations:

import

"@fhenixprotocol/contracts/utils/debug/Console.sol" ; import

{ FHE }

from

"@fhenixprotocol/contracts/FHE.sol" ;

contract

EncryptedNumberContract

{ using

EncryptedNumberLibrary

for EncryptedNumber ;

function

computeWithEncryptedNumbers (inEuint64 encryptedA , inEuint64 encryptedB)

public

{

// Perform some operations euint64 result = FHE . asEuint64 (encryptedA)

+ FHE . asEuint64 (encryptedB) ;

// DEBUG: Log the intermediate result uint256 debug_Result = FHE . decrypt (result) ; // Log the result Console . log (result) ;

// Perform more operations euint64 finalResult = result * FHE . asEuint64 (encryptedA) ;

return finalResult ; } } By strategically placing logs, developers can gain insights into the operations and transformations performed on encrypted numbers, greatly simplifying debugging and ensuring the integrity of computations.

Viewing Logs in the Localfhenix Docker Containerâ

Logging in the Localfhenix environment is directed to the Docker log output. To view these logs, follow these steps:

note Logging is not available on the Fhenix Testnet or Mainnet. It is only available in the Localfhenix development environment. If you are running Localfhenix using the Hardhat plugin, you can view the logs by running the following command:

docker logs localfhenix_hh_plugin -f 2>&1 | grep "Contract Log:" If you are running LocalFhenix using the Docker image directly, you must first identify the container name using thedocker ps command and then view the logs:

docker logs -f 2>&1 | grep "Contract Log:" Edit this page

Previous ð§ Type Conversions Next ð§âãï¸ Types and Operations