Permissions

Overview

ThePermissioned contract is an abstract Solidity contract that leverages EIP-712 standard signatures to enforce access controls. It's designed to be used by developers who require signature verification to restrict access to certain contract functions. While it can be used to restrict any kind of function, it's particularly useful for creating access-controlled view functions where data should only be visible to entities with a verified signature.

Use Cases

One of the common use cases for such access control is in scenarios where sensitive information must be retrieved from a contract but should not be publicly accessible. For example, a contract managing private user data may implement view functions which require a signature to confirm the identity of the requester. This ensures that only the user or an authorized party can access that user's data.

How to Use

To utilize thePermissioned contract, you would inherit it in your own contract and apply the custom modifiers to the functions you want to protect. For implementing access-controlled view functions, follow these steps:

1. Define a view function in your contract. For example, to retrieve sensitive data:
2. function getSensitiveData(Permission calldata perm) public view onlySender(perm) returns (string memory) {
3. // Logic to return sensitive data
4. }
5. Off-chain, the user generates a signature over their request using EIP-712 signed with their private key. This process typically involves structured data that lists the types of variables involved and their values. The result is a signature that proves the user consents to the requested operation.
6. Call the view function with the generated signature as one of the parameters. Only if the signature is verified and corresponds to themsg.sender
7. will the view function execute and return the sensitive data.

Example Scenario

Imagine a contract holding medical records. You want to create a secure method for patients to view their records:

pragma solidity ^ 0.8 .20 ;

import

"@fhenixprotocol/contracts/access/Permissioned.sol" ;

contract MedicalRecords is Permissioned

{

mapping (address

=> string)

private records ;

function

viewMedicalRecord (Permission calldata perm)

public view onlySender (perm)

returns

(string memory)

{ return records [msg . sender] ; } } The patient, after obtaining the appropriate signature using their private key, would submit it along with their request to view their records. The contract verifies the signature against the caller's address, and if it matches, returns the patient’s medical record.

danger In this example we are just showcasing the usage of permissions.string andaddress are still public data types and can be read directly from the chain!

Notes

- Permissioned view functions only allow access upon successful signature verification, enhancing contract's data privacy.
- Users need to protect their private keys used to generate EIP-712 signatures to maintain the integrity of the access control system.
- Developers must integrate off-chain EIP-712 compliant signing processes to ensure users can generate valid signatures for contract interactions.
- EIP-712 signatures provide strong assurances of user intention, making them ideal for sensitive operations.Edit this page

Previous Require Statements Next Type Conversions