To add an event listener to get data for **toNode** in the **storeData** function, you can define an event and emit it whenever data is stored. Here's an example code snippet:

pragma solidity >0.4.0;

contract data {

struct Data {

uint256 timestamp;

string dataType;

string dataHash;

address fromNode;

address toNode;

}

mapping (uint256 => Data) private dataStore;

mapping (address => bool) private authorized;

event DataStored(address indexed fromNode, address indexed toNode, uint256 indexed index, uint256 timestamp, string dataType, string dataHash);

function storeData(uint256 index, uint256 timestamp, string memory dataType, string memory dataHash, address fromNode, address toNode) public {

require(authorized[msg.sender], "Unauthorized access");

dataStore[index] = Data(timestamp, dataType, dataHash, fromNode, toNode);

emit DataStored(fromNode, toNode, index, timestamp, dataType, dataHash);

}

function getData(uint256 index) public view returns (uint256, string memory, string memory, address, address) {

require(authorized[msg.sender], "Unauthorized access");

Data memory data = dataStore[index];

return (data.timestamp, data.dataType, data.dataHash, data.fromNode, data.toNode);

}

function grantAccess(address \_address) public {

require(msg.sender == 0x8614412ED09e8e4Db8eb43025d7830aB5a768772, "Only the contract owner can grant access");

authorized[\_address] = true;

}

function revokeAccess(address \_address) public {

require(msg.sender == 0x8614412ED09e8e4Db8eb43025d7830aB5a768772, "Only the contract owner can grant access");

authorized[\_address] = false;

}

}

In this modified contract, we have added an event called **DataStored** that takes the **fromNode**, **toNode**, **index**, **timestamp**, **dataType**, and **dataHash** as parameters. We have also added an **emit** statement inside the **storeData** function that triggers the **DataStored** event whenever data is stored.

Now, to get data for a specific **toNode**, you can listen to the **DataStored** event and filter the events based on the **toNode** parameter. Here's an example code snippet to demonstrate how you can do that:

// Example code to listen for DataStored events and filter by toNode

contract MyListener {

data private dataContract;

constructor(address \_dataContract) {

dataContract = data(\_dataContract);

}

function getEventDataForToNode(address \_toNode) public view returns (uint256[] memory, string[] memory, string[] memory) {

uint256[] memory timestamps = new uint256[](100); // assuming there are at most 100 events for a given toNode

string[] memory dataTypes = new string[](100);

string[] memory dataHashes = new string[](100);

uint256 count = 0;

for (uint256 i = 0; i < 100; i++) { // assuming there are at most 100 data points in the dataStore

(uint256 timestamp, string memory dataType, string memory dataHash, address fromNode, address toNode) = dataContract.getData(i);

if (toNode == \_toNode) {

timestamps[count] = timestamp;

dataTypes[count] = dataType;

dataHashes[count] = dataHash;

count++;

}

}

return (timestamps, dataTypes, dataHashes);

}

}