Combined Solidity Smart Contracts

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

/*
====================================
Contract 1: Directed Acyclic Graph (DAG)
====================================
Implements a simple Directed Acyclic Graph structure with cycle prevention.
*/
contract ShortDAG {
    mapping(uint => uint[]) public edges;
    mapping(uint => bool) public exists;

    event Node(uint id);
    event Link(uint from, uint to);

    function addNode(uint id) public {
        require(!exists[id], "Already exists");
        exists[id] = true;
        emit Node(id);
    }

    function addEdge(uint from, uint to) public {
        require(exists[from] && exists[to], "Nodes must exist");
        require(from != to, "No self-link");
        require(!hasPath(to, from), "Cycle detected");

        edges[from].push(to);
        emit Link(from, to);
    }

    function getLinks(uint id) public view returns (uint[] memory) {
        return edges[id];
    }

    function hasPath(uint from, uint to) internal view returns (bool) {
        if (from == to) return true;
        for (uint i = 0; i < edges[from].length; i++) {
            if (hasPath(edges[from][i], to)) return true;
        }
        return false;
    }
}

/*
===============================================
Contract 2: Fabric Channel Creation (Simulation)
===============================================
Simulates the process of creating a private channel between organizations.
*/
contract FabricChannelCreator {
    event ChannelRequested(string channelName, string org1, string org2);

    function requestChannelCreation(
        string memory channelName,
        string memory org1,
        string memory org2
    ) public {
        emit ChannelRequested(channelName, org1, org2);
    }
}
```