Ping Pong

Ping is a contract on some domain andPong is a contract on some other domain. The user sends a ping and a pong will be sent back!

This example demonstrates how to use nestedxcall s and a single direction emulating "callback" behavior.

Ping Contract

ThePing contract contains an externalstartPingPong function that the user will call to initiate the flow. Italso implementsIXReceiver because it will act as the "target" of anxcall fromPong . ThexReceive function here is essentially a callback function - we can verify that something happened inPong 's domain and handle any results passed back.

```

Copy // SPDX-License-Identifier: UNLICENSED pragmasolidity^0.8.15;

import{IConnext}from"@connext/interfaces/core/IConnext.sol";
import{IXReceiver}from"@connext/interfaces/core/IXReceiver.sol";

/ *@titlePing* @noticePing side of a PingPong example. */ contractPingisIXReceiver{ // The Connext contract on this domain IConnextpublicimmutableconnext;

// Number of pings this contract has received uint256publicpings;

constructor(address_connext) { connext=IConnext(_connext); }

/ *@noticeStarts the ping pong s. equence.* @paramdestinationDomain The destination domain ID. *@paramtarget Address of the Pong contract on the destination domain.* @paramrelayerFee The fee offered to relayers. */ functionstartPingPong( addresstarget, uint32destinationDomain, uint256relayerFee )externalpayable{ require( msg.value==relayerFee, "Must send gas equal to the specified relayer fee" );

// Include the relayerFee so Pong will use the same fee // Include the address of this contract so Pong will know where to send the "callback" bytesmemorycallData=abi.encode(pings,address(this),relayerFee);

connext.xcall{value:relayerFee}( destinationDomain,// _destination: domain ID of the destination chain target,// _to: address of the target contract (Pong) address(0),// _asset: use address zero for 0-value transfers msg.sender,// _delegate: address that can revert or forceLocal on destination 0,// _amount: 0 because no funds are being transferred 0,// _slippage: can be anything between 0-10000 because no funds are being transferred callData// _callData: the encoded calldata to send ); }

/*@noticeThe receiver function as required by the IXReceiver interface.@devThe "callback" function for this example. Will be triggered after Pong xcalls back.* / functionxReceive( bytes32_transferId, uint256_amount, address_asset, address_originSender, uint32_origin, bytesmemory_callData )externalreturns(bytesmemory) { uint256_pongs=abi.decode(_callData,(uint256));

pings++; } }

```

Pong Contract

Pong will send a nestedxcall back toPing , including some information that can be acted on.

```

Copy // SPDX-License-Identifier: UNLICENSED pragmasolidity^0.8.15;

import{IConnext}from"@connext/interfaces/core/IConnext.sol";
import{IXReceiver}from"@connext/interfaces/core/IXReceiver.sol";

interfaceIPong{ functionsendPong( uint32destinationDomain, addresstarget, uint256relayerFee )externalpayable; }

/ *@titlePong* @noticePong side of a PingPong example. */ contractPongisIXReceiver{ // The Connext contract on this domain IConnextpublicimmutableconnext;

// Number of pongs this contract has received uint256publicpongs;

constructor(address_connext) { connext=IConnext(_connext); }

/ *@noticeSends a pong to the Ping contract.* @paramdestinationDomain The destination domain ID. *@paramtarget Address*
```

of the Ping contract on the destination domain. @paramrelayerFee The fee offered to relayers. */ functionsendPong(uint32destinationDomain, addresstarget, uint256relayerFee)internal{ // Include some data we can use back on Ping bytesmemorycallData=abi.encode(pongs);

connext.xcall{value:relayerFee}(destinationDomain,// _destination: Domain ID of the destination chain target,// _to: address of the target contract (Ping) address(0),// _asset: use address zero for 0-value transfers msg.sender,// _delegate: address that can revert or forceLocal on destination 0,// _amount: 0 because no funds are being transferred 0,// _slippage: can be anything between 0-10000 because no funds are being transferred callData// _callData: the encoded calldata to send); }

/ *@noticeThe receiver function as required by the IXReceiver interface.*@devThe Connext bridge contract will call this function. */ functionxReceive(bytes32_transferId, uint256_amount, address_asset, address_originSender, uint32_origin, bytesmemory_callData)externalreturns(bytesmemory) { // Because this call is* not* authenticated, the _originSender will be the Zero Address // Ping's address was sent with the xcall so it can be decoded and used for the nested xcall (uint256_pings, address_pingContract, uint256_relayerFee)=abi.decode(_callData,(uint256,address,uint256));

pongs++;

// This contract sends a nested xcall with the same relayerFee value used for Ping. That means // it must own at least that much in native gas to pay for the next xcall. require(address(this).balance>=_relayerFee, "Not enough gas to pay for relayer fee");

// The nested xcall sendPong(_origin,_pingContract,_relayerFee); }

/ *@noticeThis contract can receive gas to pay for nested xcall relayer fees.*/ receive()externalpayable{}

fallback()externalpayable{} }

```

An important note forPong is thatsendPong isnot payable and neither isxReceive . So in order for the 2ndxcall to work withrelayerFees , the someone has to send native gas on destination toPong . In practice, this can take the form of a "gas tank" mechanism that can be filled by users or subsidized by protocols.

Connext is working on an upgrade that will soon allow nested relayer fees to be deducted from the transacting asset, eliminating the need to fund receivers in their native gas token.

Previous Authenticated Greeter Next xERC20 Last updated9 months ago On this page *Ping Contract * Pong Contract

Edit on GitHub
```