Hello Wormhole Explained

In Part 1 (HelloWormhole), we wrote a fully functioning cross-chain application that allows users to request, from one contract, that aGreetingReceived event to be emitted from one of the other contracts on a different chain.

To do this, we made use of theWormhole Relayer contract (full interface ,implementation).

Specifically we used the following functions:

```

Copy

functionsendPayloadToEvm( uint16targetChain, addresstargetAddress, bytesmemorypayload, uint256receiverValue, uint256gasLimit )externalpayablereturns(uint64sequence);

functionquoteEVMDeliveryPrice( uint16targetChain, uint256receiverValue, uint256gasLimit )externalviewreturns(uint256nativePriceQuote,uint256);

```

ThisWormhole Relayer contract is deployed on both Mainnet and Testnets.

Testnet

- Chain ID 4: BSC Testnet0x80aC94316391752A193C1c47E27D382b507c93F3
- Chain ID 5: Polygon Testnet0x0591C25ebd0580E0d4F27A82Fc2e24E7489CB5e0
- Chain ID 6: Avalanche Testnet (Fuji)0xA3cF45939bD6260bcFe3D66bc73d60f19e49a8BB
- Chain ID 14: Celo Testnet:0x306B68267Deb7c5DfCDa3619E22E9Ca39C374f84
- Chain ID 16: Moonbeam Testnet0x0591C25ebd0580E0d4F27A82Fc2e24E7489CB5e0
-

Mainnet

For all Mainnet deployments, the contract has the same address:0x27428DD2d3DD32A4D7f7C497eAaa23130d894911

It is currently deployed to:

- Chain ID 2: Ethereum
- Chain ID 4: Binance Smart Chain
- Chain ID 5: Polygon
- Chain ID 6: Avalanche
- Chain ID 10: Fantom
- Chain ID 13: Klaytn
- Chain ID 14: Celo
- Chain ID 16: Moonbeam
- Chain ID 23: Arbitrum
- Chain ID 24: Optimism
-

Note: thetargetChain input should be the number in front of your desired chain; e.g. 2 for Ethereum, 4 for Binance Smart Chain, 6 for Avalanche, etc...

So for any of these chains, thesendPayloadToEvm of the deployed contract may be called.

Note: The methodsendPayloadToEvm must be called with a specificmsg.value , specifically(uint256 requiredMsgValue,) = quoteEVMDeliveryPrice(targetChain, receiverValue, gasLimit)

Calling this function will result in thereceiveWormholeMessages endpoint on address ‘targetAddress’ on chain ‘targetChain’ to be invoked with a gas limit of ‘gasLimit’ and msg.value of ‘receiverValue’, and with the payload parameter being ‘payload’.

How does the Wormhole Relayer contract cause a function call on a different blockchain?

Step 1

The Wormhole Relayer contract publishes the Delivery Instructions and pays the Delivery Provider

The methodsendPayloadToEvm works by publishing a wormhole message (simply an event containing bytes) with instructions for how to perform the delivery: the target chain, target address, receiver value, gas limit, payload, and any other necessary information

sendPayloadToEvm also thenpays a delivery provider it'smsg.value .

Delivery Providers are permissionless entities that help power the Wormhole Relayer network. If left unspecified, your delivery request will be assigned to the default delivery provider. Each delivery provider is allowed to set it's own pricing for relaying to a specific chain with a specific receiverValue and gasLimit.

Thefull Wormhole Relayer interface provides endpoints where you can specify the delivery provider you wish to use

In our scenario,

- a user of HelloWormhole calls ‘sendCrossChainGreeting’,
- which calls the Wormhole Relayer contract’s ‘sendPayloadToEvm’,
- which publishes the delivery instructions to the blockchain logs and pays the default delivery provider
-

Step 2

The Guardians create a signed VAA

The wormhole protocol, at it’s core, is publishing messages from blockchains that are then signed by a quorum of 19 entities called theGuardians to form signed VAAs.

Each Guardian watches the wormhole-connected blockchains and signs Wormhole events it sees, forming a VAA, thecore primitive of wormhole . Once a VAA obtains 13 of 19 signatures, it is considered fully signed.

In our scenario, 13 of 19 guardians sign the delivery instructions to form a signed VAA

Step 3

The Delivery Provider watches for signed VAAs containing deliveries that it has been assigned to, and (off chain!) parses the delivery instructions

The Delivery Provider is likely running some form of theRelayer Engine , watching the guardian network for signed VAAs containing wormhole messages from the Wormhole Relayer contract that indicate delivery instructions for them to execute on.

The delivery instructions can even indicate fetching other VAAs (see the sendVaasToEvm endpoint !) This feature is useful for composing with other Wormhole protocols, such as TokenBridge

The Delivery Provider parses the delivery instruction it sees and obtains the following information:

- The full signed VAA containing the delivery instructions (that it parsed)
- Any additional VAAs that were requested to be delivered
- The target chain
- The cost of the delivery (the msg.value it needs to provide, as well as the maximum possible refund it could need to pay)
-

and then calls thedeliver endpoint on the Wormhole Relayer contract on the target chain with the signed delivery instructions VAA and additional VAAs (and msg.value) as input.

In our scenario,

- The default delivery provider sees the delivery instructions VAA,
- parses the VAA off-chain to figure out the correct target chain
- and submits the VAA to the ‘deliver’ endpoint on the Wormhole Relayer contract on the target chain
-

Step 4

The Wormhole Relayer contract receives the delivery VAA, ensures the guardian signatures are valid, and calls the receiveWormholeMessages endpoint

Thedeliver endpoint on the Wormhole Relayer contract, when called by the delivery provider:

- ensures the signatures of the delivery VAA are valid
- parses the delivery instructions to figure out the targetAddress, payload, gasLimit, receiverValue, etc
- calls the ‘receiveWormholeMessages()’ endpoint of ‘targetAddress’
- with the payload, as well as additional metadata (e.g. additional VAAs, the source chain, source address, hash of the delivery VAA which can be used as a unique identifier), and with the specified gasLimit and msg.value (receiverValue)
-

A status event is then emitted to indicate whether this call succeeded or failed (and if it failed, the revert string is provided).

To see the status of your delivery requests, use the ‘getWormholeRelayerInfo’ function in the Wormhole Javascript SDK -see usage here . You can run this in HelloWormhole usingnpm run status -- --tx TRANSACTION_HASH

In our scenario, (on the target chain)

- The Wormhole Relayer contract verifies the signatures on the delivery VAA,
- and then parses the VAA to figure out the correct target address (which is our HelloWormhole contract on this chain), payload, and gas limit
- and submits the payload to the ‘receiveWormholeMessages’ endpoint of our HelloWormhole contract
- ..and then our HelloWormhole contract on the target chain does the rest!
-

Wormhole integration complete?

Let us know so we can list your project in our ecosystem directory and introduce you to our global, multichain community!

Reach out now!

Last updated1 month ago

On this page * Testnet * Mainnet * How does the Wormhole Relayer contract cause a function call on a different blockchain? * Step 1 * Step 2 * Step 3 * Step 4

Was this helpful? Edit on GitHub