Standard Relayer

?

The Standard Relayer provides a mechanism for a contract on one chain to send a message to a contract on a different chain without the developer dealing with any off chain deployments.

Currently the Standard Relayer feature is limited to EVM environments.

Find the complete list of EVM environment blockchains here .

Tutorials

- Hello Wormhole
- A tutorial that covers message passing across EVM environments
- Hello Token
- A tutorial that covers token transfer across EVM environments
-

On Chain

On chain, a smart contract interacts with theIWormholeRelayer to send and receive messages.

Sending a message

To send a message to a contract on another EVM chain, we can call thesendPayloadToEvm method, provided by theIWormholeRelayer interface.

```

Copy functionsendPayloadToEvm( // Chain ID in Wormhole format uint16targetChain, // Contract Address on target chain we're sending a message to addresstargetAddress, // The payload, encoded as bytes bytesmemorypayload, // How much value to attach to the delivery transaction uint256receiverValue, // The gas limit to set on the delivery transaction uint256gasLimit )externalpayablereturns( // Unique, incrementing ID, used to identify a message uint64sequence );

```

ThesendPayloadToEvm method is markedpayable so we can pay for our transaction to be submitted.

The value to attach to the invocation is determined by calling thequoteEVMDeliveryPrice , which provides an estimate of the cost of gas on the target chain.

```

Copy functionquoteEVMDeliveryPrice( // Chain ID in Wormhole format uint16targetChain, // How much value to attach to delivery transaction uint256receiverValue, // The gas limit to attach to the delivery transaction uint256gasLimit )externalviewreturns( // How much value to attach to the send call uint256nativePriceQuote, // uint256targetChainRefundPerGasUnused );

```

This method should be called prior to sending a message and the value returned fornativePriceQuote should be attached to the call to send the payload in order to cover the cost of the transaction on the target chain.

In total, sending a message across EVM chains can be as simple as:

```

Copy // Get a quote for the cost of gas for delivery (cost,)=wormholeRelayer.quoteEVMDeliveryPrice( targetChain, valueToSend, GAS_LIMIT );

// Send the message wormholeRelayer.sendPayloadToEvm{value:cost}( targetChain, targetAddress, abi.encode(payload), valueToSend, GAS_LIMIT );

```

Receiving a message

To receive a message using theStandard Relayer feature, the target contract must implement theIWormholeReceiver interface.

```

Copy functionreceiveWormholeMessages( bytesmemorypayload,// Message passed by source contract bytes[]memoryadditionalVaas,// Any additional VAAs that are needed (Note: these are unverified) bytes32sourceAddress,// The address of the source contract uint16sourceChain,// The Wormhole chain ID bytes32deliveryHash// A hash of contents, useful for replay protection )externalpayable;

```

The logic inside the function body may be whatever business logic is required to take action on the specific payload.

Other Considerations

Some implementation details should be considered during development to ensure safety and improve UX.

- Receiving a message from relayer
-
    - Check for expected emitter
-
    - call parseAndVerify on any additionalVAAs
- *
- Replay protection
- Message Ordering
-
    - no guarantees on order of messages delivered
- *
- Fowarding/Call Chaining
- Refunding overpayment of gasLimit
- Refunding overpayment of value sent
-

Off Chain

If taking advantage of Automatic Relaying, no off chain logic need be implemented.

While no off chain programs are required, a developer may want to track the progress of messages in flight. To track the progress of messages in flight, use the worm CLI tool'sstatus subcommand.

```

Copy wormstatusmainnetethereum0xdeadbeef

```

See theCLI tool docs for installation and usage.

See Also

Reference documentation for EVM chains is availablehere

Last updated1 month ago

On this page * Tutorials * On Chain * Sending a message * Receiving a message * Other Considerations * Off Chain * See Also

Was this helpful? Edit on GitHub