Build Executors

This document contains a high level overview of how to implement and integrate a basic third party Executor into the LayerZero V2 protocol.

Fee Quoting, Collection, and Withdrawal

Executors should implement and deploy an Executor contract on every chain they want to support. The contract must implement theILayerZeroExecutor interface, which specifies two functions:assignJob andgetFee .

interface

ILayerZeroExecutor

{ function

assignJob (uint32 _dstEid , address _sender , uint256 _calldataSize , bytes

calldata _options)

external

payable

returns

(uint256 price) ;

function

getFee (uint32 _dstEid , address _sender , uint256 _calldataSize , bytes

calldata _options)

external

view

returns

(uint256 price) ; } FunctionName Type Description assignJob Payable Called as part of_lzSend . getFee View Typically called by applications before sending the packet to estimate fees.

If your Executor is responsible for a packet, the LayerZero Endpoint will call your Executor contract'sassignJob function.

Building an Executor

The Executor is divided into two off-chain workflows: the Committer and the Executor.

Committer

1. The Committer role first listens for thePacketSent
2. event:

PacketSent (bytes encodedPacket , bytes options , address sendLibrary) ; 1. After thePacketSent 2. event, theExecutorFeePaid 3. is how you know your Executor has been assigned to commit and execute the packet.

ExecutorFeePaid (address executor , uint256 fee) ; 1. After receiving the fee, your Executor should listen for thePacketVerified 2. event, signaling that the packet can now be committed to the destination messaging channel.

PayloadVerified (address dvn , bytes header , uint256 confirmations , bytes32 proofHash) ; 1. After listening for the previous events, your Executor should perform an idempotency check by callingUltra Light Node 301 2. andUltra Light Node 302 3. :

ULN . verifiable (_packetHeader , _payloadHash) ; This function will return the following possible states:

enum

VerificationState

{ Verifying , Verifiable , Verified } If the state isVerifying , your Executor must wait for more DVNs to sign the packet's payloadHash. After a DVN signs the payloadHash, it will emitPayloadVerified .

PayloadVerified (address dvn , bytes header , uint256 confirmations , bytes32 proofHash) ; tip Your Executor only needs to perform subsequent checks ofVerificationState when it hearsPayloadVerified on the destination chain.

If the state isVerifiable , then your Executor must callcommitVerification :

function

commitVerification (bytes

calldata _packetHeader ,

bytes32 _payloadHash)

external ; If the state isVerified , the commit has already occurred and the commit workflow can be terminated.

tip To know your workflow is finished, your Executor should perform an idempotency check at the end of the commit workflow.

Executor

1. The Executor role first listens for thePacketSent
2. event:

PacketSent (bytes encodedPacket , bytes options , address sendLibrary) 1. After thePacketSent 2. event, theExecutorFeePaid 3. is how you know your Executor has been assigned to commit and execute the packet.

ExecutorFeePaid (address executor , uint256 fee) ; 1. After receiving the fee, your Executor should listen for thePacketVerified 2. event, signaling that the packet can now be executed. 3. After listening for the previous events, your Executor should perform an idempotency check:

endpoint . executable (_packetHeader , _payloadHash) This function will return the following possible states:

enum

ExecutionState

{ NotExecutable , Executable , Executed } If the state isNotExecutable , your Executor must wait for the committer to commit the message packet, or you may have to wait for some previous nonces.

If the state isExecutable , your Executor should decode the packet's options using theoptions.ts package and call the Endpoint'slzReceive function with the packet information:

endpoint . lzReceive (_origin , _receiver , _guid , _message , _extraData) tip To know your workflow is finished, your Executor should perform an idempotency check at the end of the execute workflow.

If the state isExecuted , your Executor has fulfilled its obligation, and you can terminate the Executor workflow.

Mock Executor

BothPaladin Blockchain Security andLazer Technologies have built an implementation and open-sourced the codebase for anyone interested in reviewing a sample Executor implementation.

caution These codebases are not owned by LayerZero. Exercise caution when interacting with any third party contracts or sample materials. Edit this page

Previous Build DVNs Next Support