# Create your own Hook & ISM

Hooks and ISMs have a complementary relationship: you can customize your behavior from origin and they use a pairwise ISM contract on the destination to verify your custom hook behavior.

You can implement and utilize your own hook and ISM pattern as per your requirements. You can use an external bridge provider like Wormhole or Chainlink's CCIP by implementing theIPostDispatchHook interface on the source chain andIInterchainSecurityModule on the destination chain.

IPostDispatchHook Interface * Solidity

interface

IPostDispatchHook

{ enum

Types

{ UNUSED , ROUTING , AGGREGATION , MERKLE_TREE , INTERCHAIN_GAS_PAYMASTER , FALLBACK_ROUTING , ID_AUTH_ISM , PAUSABLE , PROTOCOL_FEE , LAYER_ZERO_V1 , RATE_LIMITED , ARB_L2_TO_L1 , OP_L2_TO_L1 }

/* *@notice Returns an enum that represents the type of hook*/ function

hookType ( )

external

view

returns

( uint8 ) ;

/* *@notice Returns whether the hook supports metadata * @param metadata metadata * @return Whether the hook supports metadata*/ function

supportsMetadata ( bytes

calldata metadata )

external

view

returns

( bool ) ;

/* *@notice Post action after a message is dispatched via the Mailbox * @param metadata The metadata required for the hook * @param message The message passed from the Mailbox.dispatch() call* / function

postDispatch ( bytes

calldata metadata , bytes

calldata message )

external

payable ;

/* *@notice Compute the payment required by the postDispatch call * @param metadata The metadata required for the hook * @param message The message passed from the Mailbox.dispatch() call * @return Quoted payment for the postDispatch call* / function

quoteDispatch ( IInterchainSecurityModule Interface * Solidity

interface

IInterchainSecurityModule

{ enum

Types

{ UNUSED , ROUTING , AGGREGATION , LEGACY_MULTISIG , MERKLE_ROOT_MULTISIG , MESSAGE_ID_MULTISIG , NULL ,

// used with relayer carrying no metadata CCIP_READ , ARB_L2_TO_L1 , WEIGHTED_MERKLE_ROOT_MULTISIG , WEIGHTED_MESSAGE_ID_MULTISIG , OP_L2_TO_L1 }

/* *@notice Returns an enum that represents the type of security model * encoded by this ISM. * @dev Relayers infer how to fetch and format metadata.* function

moduleType ( )

external

view

returns

( uint8 ) ;

*/ * @notice Defines a security model responsible for verifying interchain * messages based on the provided metadata. * @param _metadata Off-chain metadata provided by a relayer, specific to * the security model encoded by the module (e.g. validator signatures) * @param _message Hyperlane encoded interchain message * @return True if the message was verified* / function

verify ( info Hooks currently expect metadata to be formatted with theStandardHookMetadata library . You can also inherit from ourAbstractMessageIdAuthorizedIsm which allows for access control for an intermediateverifyMessageId function call which sets in storage the messageId to true if received from the authorizedAbstractMessageIdAuthHook hook. This pattern is used currently in theOpStackHook <>OpStackIsm pattern.

### Workflow

### Interface

After implementing the above interfaces, you can override default hook along the hook metadata by using the overloadeddispatch call in our mailbox:

- Solidity

* @param destinationDomain Domain of destination chain * @param recipientAddress Address of recipient on destination chain as

bytes32 * @param messageBody Raw bytes content of message body * @param metadata Metadata used by the post dispatch hook * @param hook Custom hook to use instead of the default * @ return The message ID inserted into the Mailbox's merkle tree * /

### Examples

origin:alfajores * alfajores * arbitrumsepolia * arcadiatestnet * basesepolia * berabartio * bsctestnet * camptestnet * citreatestnet * ecotestnet * formtestnet * fuji * holesky * hyperliquidevmtestnet * odysseytestnet * optimismsepolia * plumetestnet * polygonamoy * scrollsepolia * sepolia * soneiumtestnet * sonictestnet * suavetoliman * superpositiontestnet * tangletestnet * unichaintestnet destination:arbitrumsepolia * alfajores * arbitrumsepolia * arcadiatestnet * basesepolia * berabartio * bsctestnet * camptestnet * citreatestnet * ecotestnet * formtestnet * fuji * holesky * hyperliquidevmtestnet * odysseytestnet * optimismsepolia * plumetestnet * polygonamoy * scrollsepolia * sepolia * soneiumtestnet * sonictestnet * suavetoliman * superpositiontestnet * tangletestnet * unichaintestnet body: * Solidity * CosmWasm * Sealevel

// send message from alfajores to arbitrumsepolia TestRecipient IMailbox mailbox =

IMailbox ( "0xEf9F292fcEBC3848bF4bB92a96a04F9ECBb78E59" ) ; IPostDispatchHook merkleTree =

IPostDispatchHook ( "0x221FA9CBaFcd6c1C3d206571Cf4427703e023FFa" ) ; mailbox . dispatch ( 421614 , "0x000000000000000000000000006489d13AcAd3B8dce4c5B31f375DE4f9451E7b38" , bytes ( "Hello, world" ) , "0x" ,

// empty metadata merkleTree ) ; Coming soon! Coming soon! * On the source calls your custom hook viaAbstractMessageIdAuthHook.postDispatch() * * . * * _postDispatch * * checks whetherlatestDispatchedId * * is the id being dispatched from the hook to make the mailbox is the contract calling the hook (since callingpostDispatch * * isn't access controlled) * * _sendMessageId * * calls your custom external bridge logic like calling the CCIP router contract. * On the destination chain, * * the external bridge will callverifyMessageId * * function (which is access-controlled) and sets themessageId * * in theverifiedMessages * * mapping to true. * * on receiving the message for the relayer, the mailbox will call your ISM contract (specified in your recipient address) which checks if the messageId in theverifiedMessages * * mapping is true and returns true to the mailbox and vice versa.

warning AbstractMessageIdAuthorizedIsm can sendmsg.value throughpostDispatch calls and we utilize theverifiedMessages ' little endian 255 bits for storing themsg.value and the top bit for the actual receipt of the messageId delivery. Therefore, you can send upto 2^255 amount of value of the native token from origin and the destination ISM can only receive 2^255 amount of value of native token on the destination chain.

### Access Control

IfpostDispatch must only be called with amessage that wasjust dispatched, thelatestDispatchedId function on the Mailbox can be used to verify the message wasactually dispatched.

info This is used instead of somerequire(mailbox == msg.sender) to supportcomposition where a hook may pass amessage along to another hook. * Solidity

The following utility is provided in theMailboxClient library for convenience.

_transferOwnership ( _owner ) ; }Edit this page Previous How to Transfer Ownership of Your Hyperlane Warp RouteNext Deploy Hyperlane with Local Agents