Axelar GMP Developer Tutorial

Learn how to use Axelar General Message Passing in order to send messages between EVM chains and Secret Network In this tutorial, you will learn how to use Axelar GMP to send astring between Polygon and Secret testnets . To learn more about the flow architecture,see the Axelar documentation here .

Prerequisites

- For GMP to work, both chain A and chain B must be EVM or Cosmos with a deployed Axelar Gateway contract. For this tutorial we will be using thePolygon testnet Axelar Gateway contracts
- . We will go into this in more detail momentarily.
- HaveMetamask installed
- , Polygon Mumbai Network added to your wallet, and your wallet funded with Polygon tokens.
-

Add Polygon Mumbai Network to your Metamask wallet by navigating to thePolygon testnet explorer and clicking "Add Mumbai Network" in the bottom righthand corner of the homepage.

Upload a contract to Polygon

In order to execute a smart contract on Secret network, you must first upload and instantiate a smart contract on Polygon that can execute messages using Axelar GMP. Axelar has agithub repository of example contracts that can be used for GMP. We will be uploading theSendReceive.sol contract to Polygon for this demo.

The Cosmwasm smart contracts in this Axelar repository are not compatible with Secret Network out of the box due to dependency issues. However, we will modify the SendReceive contract to be compatible with Secret Network. To upload the contract, we will useRemix Online IDE , which is a powerful toolset for developing, deploying, debugging, and testing Ethereum and EVM-compatible smart contracts.

First, navigate to Remix and create a new blank workspace:

Remix workspace Next, create a new file calledSendReceive.sol and paste theAxelar GMP solidity code . This will autofill your workspace with the necessary dependencies for your SendReceive.sol contract

Now all that's left is tocompile and upload the contract . Navigate to the Solidity compiler using the sidebar and click"Compile SendReceive.sol" . Then, navigate to "Deploy and run transactions." Toggle the Environment from "Remix VM (Shanghai)" to "Injected Provider - MetaMask" and make sure that in your MetaMask walletyou have currently selected Mumbai network .

Theconstructor of SendReceive.sol contains 3 variables that you must now input in order to instantiate the contract and link it to Axelar's Polygongateway contract andgas receiver contract , as well as the Polygon Mumbaichain name :

```

Copy GATEWAYCONTRACT:"0xBF62ef1486468a6bd26Dd669C06db43dEd5B849B"
GASRECEIVERCONTRACT:"0xbE406F0189A0B4cf3A05C286473D23791Dd44Cc6" CHAINNAME:"Polygon"

```

Input these strings like so and then click "Transact":

SendReceive constructor Upon successful instantiation, the contract address will be returned in the Remix terminal, which you can thenview on Polygonscan . And the deployed contract can now be interacted with in the "Deployed Contracts" window:

Congrats, you've just deployed an Axelar GMP-compatible contract to Polygon testnet that can send and receive messages to and from a Secret Network smart contract

Upload a contract to Secret Network

Now that you've uploaded a GMP-compatible contract to Polygon, let's do the same on Secret Network so that the contracts can communicate with each other across the Cosmos!

First, clone this Secret Network examples repository:

```

Copy gitclonehttps://github.com/scrtlabs/examples

```

Then,cd into theexamples/secret-ethereum-gmp folder

```

Copy cdexamples/secret-ethereum-gmp

```

and compile the contract by runningmake build-mainnet in your terminal.

```

Copy makebuild-mainnet

```

If this is your first time working with a Secret contract, visit theGetting Started docs to properly configure your developer environment. Now, open a new terminal window andcd intoexamples/secret-ethereum-gmp/node

```

Copy cdexamples/secret-ethereum-gmp/node

```

and then runnpm install to install thepackage.json dependencies.

```

Copy npminstall

```

Create a.env file inexamples/secret-ethereum-gmp/node and add your wallet mnemonic in order to upload the contract:

.env config You can thenupload and instantiate the contract by runningnode index.js .

```

Copy nodeindex.js

```

Upon successful instantiation, a Secret contract address is returned that you can then use to send messages to and from Polygon:

Secret contract address upon successful instantation Now let's send astring from Polygon to Secret!

Send a string from Polygon to Secret

Now that you have a GMP-compatible contract instantiated on Secret Network, you have all of the variables needed in order to send a cross-chain message using theSendReceive.sol contract.

In order to send messages using Axelar GMP,the user prepays the relayer gas fee on the source chain to Axelar's Gas Services contract .

You can do this in Remix by navigating to the "Deploy and run transactions" tab in the sidebar and adding gas to prepay the gas fee. To make sure you have enough gas, add .40 Matic, (roughly .20 USD or 400000000000000000Wei), to the transaction:

Axelar Gas fee Now all that's left is to execute the transaction! From the "Deployed contracts" section of Remix, open the dropdown for thesend function, which should have three inputs:destinationChain ,destinationAddress , andmessage . Input the following:

```

Copy destinationChain:"secret" destinationAddress:"Your Secret Contract Address" destinationMessage:"Your message that you want to send!"

```

Once you have inputed these strings and your contract address, select "transact." Congratulations! You've just sent astring from Polygon to Secret Network!

- Use Polygonscan to track the transaction on Polygon
- . Here is an alreadydeployed contract for reference

- . Transaction times vary depending upon Poygon network congestion; you might want to "speed up" the transaction in Metamask.
- Use Axelarscan to track the transaction on Axelar
- . Here is asuccessful transaction for reference
- . *

Query the string on Secret

Now that you've successfully executed a cross-chain message from Polygon to Secret using Axelar GMP, let's query the message on Secret Network to see if the message was actually received by the Secret contract.

First, confirm that the transaction has been successfully relayed by Axelar .

The Axelarscan status diagram indicates the following 5 steps were executed successfully: Axelarscan Transaction Status Once the transaction has been executed successfully, you can useSecret.js to query the message:

```

Copy letget_stored_message=async()=>{ letquery=awaitsecretjs.query.compute.queryContract({ contract_address:contractAddress, query:{ get_stored_message:{}, }, code_hash:contractCodeHash, });

console.log(query); };

get_stored_message();

```

To execute this query, navigate to thequery.js file inexamples/secret-ethereum-gmp/node and replace thecontractAdress andcontractCodeHash with your contract address and code hash, respectively.

Then runnode query .

```

Copy nodequery

```

If the message was executed successfully, the query will return theEthereum wallet address that sent the transaction as well as themessage that the wallet included:

```

Copy { sender:0x49e01eb08bBF0696Ed0df8cD894906f7Da635929, message:one small step for Secret! }

```

Great work! Now let's send a string from Secret to Polygon!

Send a string from Secret to Polygon

To execute theSendMessageEvm transaction, navigate to theexecute.js file inexamples/secret-ethereum-gmp/node and replace thecontractAdress andcontractCodeHash with your contract address and code hash.

Then, updatedestinationAddress to your Polygon contract address, andmyMessage to the message that you want to send:

execute.js variables

Gas Token

Next, in order to send a GMP message from Secret to Polygon, you need to acquire some AXL tokens and include the correct IBC denom representing those tokens to your transaction to pay for gas, so that the message can be executed over IBC

Learn more about IBC denomshere . Thecorrect IBC denom is already included in the secret.js transaction, but in order for it to execute successfully,you need to have this IBC denom funded in your wallet . To add this token to your wallet, you can send Axelar tokens to your Secret wallet address over IBC.

If this is your first time sending Axelar testnet tokens over IBC, see an Axelar Gas Token video tutorialhere , which guides you through the following steps: 1. Add Axelar testnet to your Keplr wallet by visitingAxelar Satellite 2. (upon visiting the website, a Keplr notification will pop up that says,"https://testnet.satellite.money would like to add blockchain axelar-testnet-lisbon-3 to Keplr" 3.) 4. Procure AXL testnet tokens from theAxelar faucet 5. . 6. Send AXL testnet tokens to your Secret testnet wallet address over IBC usingAxelar Satellite 7. . 8.

Upon successful transfer, you should now see the new AXL IBC token in your Keplr Wallet:

AXL IBC token Once you have properly configured yourexecute.js file and procured the IBC denom needed to execute the transaction, all that's left is to runnode execute.

```

Copy nodeexecute

```

The transaction should return atransactionHash as well as data about the IBC routing:

send_message_evm() transaction Now, navigate toAxelarscan to monitor the status of the transaction.

And for good measure,view the transaction on Polygonscan to see that the message was received!

To actually see the message that you sent, click on "Click to see more":

Polygonscan transaction And then at "Input Data",view the input as UTF-8 to reveal the decoded message!

Decoded String sent from Secret to Ethereum Congratulations! You now have all of the tools to send a message from Secret Network to Polygon using Axelar GMP!

Summary

In conclusion, Axelar's General Message Passing (GMP) offers a powerful solution for achieving secure interchain communication and token transfers across diverse blockchain ecosystems, including Cosmos and Ethereum. With GMP, developers can seamlessly execute smart contracts on one blockchain from another, fostering complete composability within the Web3 landscape. This documentation has guided you through the process of deploying GMP-compatible contracts on both Polygon and Secret Network, illustrating how to send messages across these networks. By following these steps, you have unlocked the potential for seamless interoperability and enhanced functionality in your blockchain applications. Axelar GMP paves the way for a more interconnected and efficient blockchain ecosystem. Congratulations on your successful journey with Axelar GMP!

If you run into any errors or questions, ping the#dev-issues channel on Secret Network's Discord and somebody will get back to you shortly 😊

Last updated1 month ago On this page *Prerequisites * Upload a contract to Polygon * Upload a contract to Secret Network * Send a string from Polygon to Secret* Query the string on Secret * Send a string from Secret to Polygon* Summary

Was this helpful? Edit on GitHub Export as PDF