Gateway

Overview

Wormhole Gateway is a Cosmos-SDK chain that provides a way to bridge non-native assets into the Cosmos ecosystem and serves as a source for unified liquidity across Cosmos chains.

Because IBC is used to bridge assets from Gateway to Cosmos chains, liquidity fragmentation is avoided and liquidity for foreign assets bridged via Wormhole into Cosmos is unified across Cosmos chains. In addition to facilitating asset transfers,Wormhole Gateway (FKAwormchain , AKAShai-Hulud) allows Wormhole to ensure proper accounting with theaccountant .

Details

Wormhole Gateway is implemented as a set of contracts and modules.

The contract addreses for these components are:

Contract Mainnet Address Testnet Address Wormhole core bridge
wormhole1ufs3tlq4umljk0qfe8k5ya0x6hpavn897u2cnf9k0en9jr7qarqqaqfk2j
wormhole16jzpxp0e8550c9aht6q9svcux30vtyyyyxv5w2l2djjra46580wsazcjwp Wormhole token bridge
wormhole1466nf3zuxpya8q9emxukd7vftaf6h4psr0a07srl5zw74zh84yjq4lyjmh
wormhole1aaf9r6s7nxhysuegqrxv0wpm27ypyv4886medd3mrkrw6t4yfcnst3qpex IBC Translator
wormhole14ejqjyq8um4p3xfqj74yld5waqljf88fz25yxnma0cngspxe3les00fpjx
wormhole1ctnjk7an90lz5wjfvr3cf6x984a8cjnv8dpmztmlpcq4xteaa2xs9pwmzk

Wormhole Core Contracts

Thecore contracts to emit messages and verifyGuardian signatures are still required on each Cosmos chain that requires generic message passing. Notably, for Gateway token bridging, no core contracts need be deployed.

IBC Shim Contract

A CosmWasm contract that handles bridging into and out of the Cosmos ecosystem by translating between Wormhole and IBC message formats. It maintains a mapping ofchainId -> channelId for whitelisted IBC channels to send packets over and accept packets from.

The contract supports transfersinto the Cosmos ecosystem by receivingContract Controlled Transfer VAAs .

The logical flow of this type of transfer is as follows:

- Redeem the VAA against theToken Bridge
- MintToken Factory
- tokens
- Decode the additional payload as aGatewayIbcTokenBridgePayload
- Send tokens via IBC to destination cosmos chains
-

The contract also supports transfersout of the Cosmos ecosystem by implementing anexecute handler which accepts aGatewayIbcTokenBridgePayload and an amount of tokenfactory tokens ininfo.funds (which are the tokens to be bridged out).

The logical flow for this type of transfer is as follows:

- Burn theToken Factory
- tokens
- Unlock the CW20 tokens
- Grant approval to theToken Bridge
- to spend the CW20 Tokens
- CallInitiateTransfer
- orInitiateTransferWithPayload
- based on whether theGatewayIbcTokenBridgePayload
- is of typeSimple
- orContractControlled
-

Token Factory Module

A deployment of the canonicalToken Factory module on Wormhole Gateway to create new tokens.

IBC Composability Middleware

The IBC Composability Middleware sits on top of thePFM (Packet Forwarding Module) and IBC Hooks middleware to compose the two together. It enables integrators on Cosmos chains to support both theCosmos -> Cosmos andCosmos -> External flows with a single payload structure.

It accepts a payload ofGatewayIbcTokenBridgePayload and determines whether to call the PFM or IBC Hooks middleware by looking up thechainId in the payload.

1. If thechainId
2. is an IBC-enabled chain, it formats a payload for the PFM to forward the ICS20 transfer to the IBC-enabled destination chain.
3. If thechainId
4. is an external chain, it will format a payload for the IBC Hooks middleware to call the IBC Shim contract'sexecute
5. handler to bridge out.
6.

IBC Hooks Middleware

A deployment of theIBC Hooks Middleware on Wormhole Gateway allows ICS-20 token transfers to also initiate contract calls.

Integration

Integration with Wormhole Gateway can be accomplished with a few lines of code and supports

- Transfers from anExternal Chain
- to any supportedCosmos Chain
- , seeInto Cosmos
- Transfers from any supportedCosmos Chain
- to anExternal Chain
- , seeOut of Comsos
- Transfers between any supportedCosmos Chains
- , seeBetween Cosmos Chains
-

Into Cosmos

To bridge assets into a Cosmos chain, an asset transfer is initiated on the foreign chain with apayload that is understood by the Gateway, or more specifically, theIBC Shim Contract .

Once received on the Gateway, the asset's CW20 representation is sent to the destination chain through IBC using the well establishedICS20 protocol .

An example using theSDK :

```

Copy import*aswhfrom'@certusone/wormhole-sdk';

// ...

consttransferDetails={ gateway_transfer:{// This is a simple transfer, no additional payload chain:4000,// Chain Id of the Cosmos chain we're sending to recipient:"",// Address of recipient (base64 encoded bech32) fee:0,// Fee for transfer (0 for now) nonce:0, } }

constibcTranslatorAddress="wormhole14ejqjyq8um4p3xfqj74yld5waqljf88fz25yxnma0cngspxe3les00fpjx" // Convert the transfer details to a Uint8Array for sending constpayload=newUint8Array(Buffer.from(JSON.stringify(transferDetails)))

// Send transfer transaction on Ethereum awaittxReceipt=wh.transferFromEth( wh.consts.TESTNET.eth.token_bridge// source chain token bridge address wallet,// signer for eth tx "0xdeadbeef...",// address of token being transferred 10000000n,// amount of token in its base units wh.consts.CHAINS.wormchain,// chain id we're sending to ibcTranslatorAddress,// The address of the ibc-translator contract on the Gateway 0,// relayer fee, 0 for now {},// tx overrides (gas fees, etc...) payload// The payload Gateway uses to route transfers );

// ...

```

Out of Cosmos

To bridge assets out of the Cosmos ecosystem or between Cosmos chains, an IBC transfer is initiated on the source chain to the Gateway with a payload containing details about the transfer in thememo field.

For example, usingcosmjs :

```

Copy constwallet=awaitDirectSecp256k1HdWallet.fromMnemonic(faucet.mnemonic);
constclient=awaitSigningStargateClient.connectWithSigner( simapp.tendermintUrl, wallet, defaultSigningClientOptions );

constmemo=JSON.stringify({ gateway_ibc_token_bridge_payload:{ gateway_transfer:{ chain:0,// chain id of receiver recipient:"",// address of receiver fee:0,// fee to cover transfer nonce:0,// } } })
constibcTranslatorAddress="wormhole14ejqjyq8um4p3xfqj74yld5waqljf88fz25yxnma0cngspxe3les00fpjx"
constresult=awaitclient.sendIbcTokens( faucet.address0,// sender address ibcTranslatorAddress,// receiver address coin(1234,"ucosm"),// amount and coin "transfer",// source port "channel-2186",// source channel, TODO: fill in once deployed timeoutHeight,// timeoutTimestamp,// 0,// fee to cover transaction memo// formatted payload with details about transfer );

```

Between Cosmos Chains

Transfers between Cosmos chains work exactly the same asbridging out of Cosmos from an implementation perspective. The exception being that thechain id passed is a Cosmos chain.

Datastructures

Core datastructures that are used by the Gateway protocol.

GatewayIbcTokenBridgePayload

The core datastructure of Gateway token transfers is theGatewayIbcTokenBridgePayload , containing details about the transfer that the Gateway uses to perform the transfer.

```

Copy pubenumGatewayIbcTokenBridgePayload{ GatewayTransfer{ chain:u16, recipient:Binary, fee:u128, nonce:u32, }, GatewayTransferWithPayload{ chain:u16, contract:Binary, payload:Binary, nonce:u32, }, }

```

When sending aGatewayIbcTokenBridge payload, it must be serialized as json.

For a proper json, encoding TheBinary values are base64 encoded.

Therecipient for cosmos chain chains are base64 encoded bech32 addresses. For example, if therecipient iswormhole1f3jshdsmzl03v03w2hswqcfmwqf2j5csw223ls , the encoding will be the direct base64 encoding ofd29ybWhvbGUxZjNqc2hkc216bDAzdjAzdzJoc3dxY2Ztd3FmMmo1Y3N3MjIzbHM= .

Thechain values map toWormhole chain IDs .

Thefee andnonce are Wormhole-specific parameters, both of which are unused today.

For incoming IBC messages from Cosmos/IBC chains, thereceiver field will be base64 encoded in theSimple.recipient field, and thechannel-id will be included as the equivalent wormholechain id.

Fee Structure

The fees for using Gateway are minimal. At the moment, the source chain gas is the only cost.

Fees Required

- Source Chain Gas
- : Gas fees on the source chain (e.g. Ethereum) must be covered.
- Relayer Fee [Source Chain => Gateway]
- : The cost for a wormhole message to be processed. This is currently0
- but may change in the future.
- Destination Chain Gas [Non Cosmos]
- : Gas fees on a destination chain (e.g. Ethereum) must be covered by either the relayer or, in the case of manual redemption, the user.
-

Fees Not Required

- Gateway
- : Gateway doesn't have token-priced metering or require gas fees to be paid by the user.
- Relayer Fee [Gateway => Cosmos]
- : Relayers aren't incentivized by user fees.
- Destination Chain [Cosmos]
- : IBC relayers cover the processing cost on the destination chain.
-

See Also

Gateway Block Explorer

Wormhole Gateway is, of course, open source and the source is available here

The contracts that make this possible are available here

Last updated1 month ago

On this page * Overview * Details * Integration * Datastructures * Fee Structure * See Also

Was this helpful? Edit on GitHub