Core Contracts

Core Contract

The Core Contracts are the mechanism by which all Wormhole messages are emitted. All cross chain applications either interact directly with the Core Contract or interact with another contract that does. There is one Core Contract on each blockchain in the ecosystem, and this is the contract which the Guardians are required to observe. The Core Contract is the contract that ultimately emits the messages theGuardians pick up as anObservation .

In general, Core Contracts are simple and can be broken down to asending andreceiving side, which we'll define next.

Sending

Currently there are no fees to publish a message (with the exception of publishing on Solana) but this is not guaranteed to always be the case in the future.

The implementation strategy for publishMessage differs by chain, but the general strategy consists of the Core Contract posting the emitterAddress (the contract which called publishMessage), sequenceNumber, and consistencyLevel into the blockchain logs. Once the desired consistencyLevel has been reached and the message passes all of the Guardians' optional checks, the Guardian Network will produce the requested VAAs.

The method signature for publishing messages

```

Copy publishMessage( intnonce, byte[] payload, intconsistencyLevel )returnsint sequenceNumber

```

Parameters

payload

The content of the emitted message, an arbitrary byte array. It may be capped to a certain maximum length due to the constraints of individual blockchains.

consistencyLevel

Some advanced integrators may want to get messagesbefore finality, which is where theconsistency_level field offers chain-specific flexibility. Theconsistency_level can be considered as a numericenum data type where the value is treated differently for different chains.

It describes the level of finality to reach before the guardians will observe and attest the emitted event. This is a defense against reorgs and rollbacks since a transaction, once consideredfinal , is guaranteed not to have the state changes it caused be rolled back.

Different chains use different consensus mechanisms, so there are different finality assumptions with each one, see the options for finality in theEnvironments pages.

nonce

A free integer field that can be used however the developer would like. Note that a differentnonce will result in a different digest.

Returns

sequenceNumber

A unique number that increments for every message for a given emitter (and implicitly chain). This combined with the emitter address and emitter chain ID allows the VAA for this message to be queried from theAPIs

Receiving

The method signature for receiving messages, encoded as a VAA

```

Copy parseAndVerifyVAA(byte[] VAA)

```

When passed a VAA, this function will either return the payload and associated metadata for the VAA or throw an exception. An exception should only ever throw if the VAA fails signature verification, indicating the VAA is invalid or inauthentic in some form.

A developer should take care to make sure this method is called during the execution of a transaction where a VAA is passed to ensure the signatures are checked and verified.

Multicast

Please notice that there is no destination address or chain in these functions.

VAAs simply attest that "this contract on this chain said this thing." Therefore, VAAs are multicast by default and will be verified as authentic on any chain they are brought to.

This multicast-by-default model is integral to the design. Having this multicast capacity makes it easy to synchronize state across the entire ecosystem, because a single blockchain can make its data available to every chain in a single action with low latency. This reduces the complexity of the n^2 problems encountered by routing data to a large number of blockchains.

This does not mean an applicationcannot specify a destination address or chain. For example the Token Bridge and Standard Relayer contracts require that some destination details are passed and verified on the destination chain.

Because the VAA creation is separate from relaying, there isno additional cost to the multicast model when a single chain is being targeted. If the data isn't needed on a certain blockchain, don't relay it there, and it won't cost anything.

Other provided contracts

Token Bridge

Before a token transfer can be made, the token being transfered must exist as a wrapped asset on the target chain. This is done byAttesting the token details on the target chain. The Token Bridge contract allows token transfers between blockchains through a lock and mint mechanism, using theCore Contract with aspecific payload to pass information about the transfer.

The Token Bridge also supports sending tokens with some additional data in the form of arbitrary byte payload attached to the token transfer. This type of transfer is referred to as aContract Controlled Transfer .

While theCore Contract has no specific receiver by default, transfers sent through the token bridgedo have a specific receiver chain and address to ensure the tokens are minted to the expected recipient.

NFT Bridge

The NFT Bridge functions similarly to theToken Bridge but with special rules for what may be transferred and how the wrapped version is created on the destination chain.

Last updated1 month ago

On this page * Core Contract * Sending * Receiving * Multicast * Other provided contracts * Token Bridge * NFT Bridge

Was this helpful? Edit on GitHub