**Onboarding Xode to Basilisk**

This guide provides a detailed step-by-step process for onboarding tokens to the Basilisk parachain. Each step is explained with precision to ensure proper execution and successful integration of the XON token. Follow the instructions carefully.

**Note:**

As per the Basilisk documentation, both parachain sovereign accounts (Xode and Basilisk) must have enough funds (approximately 10.1 KSM) on the relay chain to reserve a deposit for HRMP channels and to process XCM messages. Since Basilisk already has the necessary KSM, we only need to ensure that Xode has sufficient funds to proceed with the channel creation.

**Step 1: Create an Encoded Hash in the Relay Chain**

Initiate an open channel request in the Kusama relay chain to establish communication with the Basilisk parachain. Use the hrmpInitOpenChannel function under the hrmp module to create the connection. Once executed, an encoded hash will be generated, which will be used in the subsequent steps.

Extrinsic: hrmp

Function: hrmpInitOpenChannel

Parameters:

* recipient: 2090 — This should be the parachain ID of the target chain, which in this case is Basilisk.
* proposedMaxCapacity: 1000 — The default configuration for HRMP (Horizontal Relay Message Passing) channels in the Kusama relay chain.
* proposedMaxMessageSize: 102400 — The default message size for HRMP channels in the Kusama relay chain.

Upon encoding, an encoded hash **0x3c002a080000e803000000900100** will be generated, which will be used in Step 2 to facilitate the communication between the Kusama relay chain and the Basilisk parachain.

**Step 2: Execute the Encoded Hash on the Xode Parachain**

Take the encoded hash generated in Step 1 and prepare to execute it on the Xode parachain using the send function under the polkadotXcm module. This step involves preparing the extrinsic with parameters including the destination (v3 for Xode), assets to withdraw, execution purchase details, and other necessary transaction settings. The key parameter here is the encoded hash **0x3c002a080000e803000000900100**, which will facilitate the connection between the relay chain and Basilisk.

Extrinsic: polkadotXcm

Function: send

Parameters:

* dest (v3):
  + parent: 0
  + interior: Here
* message (v3):
  + WithdrawAsset:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + BuyExecution:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + Transact:
    - originKind: Native
      * refTime: 1000000000
      * proofSize: 65536
    - call:
      * encoded: 0x3c002a080000e803000000900100
  + RefundSurplus
  + DepositAsset:
    - assets: Wild
      * Wild: AllCounted
      * AllCounted: 1
    - beneficiary:
      * parent: 0
      * interior: (X2, (AccountId32(5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY)))

**Step 3: Create a Referenda on Basilisk**

On Basilisk, create a referenda to manage the channel request from Xode and register the XON token. To do this, first prepare a pre-image that includes a batch call using the batchAll function under the utility pallet. The batch call should include the following extrinsics:

* Two **polkadotXcm** extrinsics: One to accept the open channel request from Xode and another to establish further communication.
* The **assetRegistry**.register extrinsic to register the XON token on Basilisk.

Once the pre-image is created, proceed to create a referenda, using the pre-image hash as a parameter. This referenda will be used for voting on the proposal, in line with Polkadot’s OpenGov process.

The referenda will not be executed immediately but will be voted on by the governance participants. Once approved, these extrinsics will be executed:

* **polkadotXcm.send** (First request to accept Xode’s open channel)
* **polkadotXcm.send** (Second request for further communication)
* **assetRegistry.registe**r (Register the XON token on Basilisk)

**Step 3.1: Accept Xode’s Open Channel Request**

To accept the open channel request from Xode, prepare the extrinsic using the hrmpAcceptOpenChannel function under the hrmp module in the Kusama relay chain. This extrinsic will generate an encoded hash that will be used in Step 3.2.

Extrinsic: hrmp

Function: hrmpAcceptOpenChannel

Parameters:

* sender: 3344 — The parachain ID of Xode.

The resulting encoded hash will be 0x3c01100d0000, which should be saved for use in Step 3.2.

**Step 3.2: Prepare an XCM Message to Accept Xode’s Open Channel Request**

Prepare an XCM message to accept Xode's open channel request by using the send function under the polkadotXcm module. The message should include the encoded hash 0x3c01100d0000, indicating acceptance of the request from Basilisk.

This message is only prepared at this stage, and it will be sent once the referenda is approved.

Extrinsic: polkadotXcm

Function: send

Parameters:

* dest (v3):
  + parent: 0
  + interior: Here
* message (v3):
  + WithdrawAsset:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + BuyExecution:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + Transact:
    - originKind: Native
      * refTime: 1000000000
      * proofSize: 65536
    - call:
      * encoded: 0x3c01100d0000
  + RefundSurplus
  + DepositAsset:
    - assets: Wild
      * Wild: AllCounted
      * AllCounted: 1
    - Beneficiary:
      * parent: 0
      * interior: (X2, (AccountId32(5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY)))

**Step 3.3: Send an Open Channel Request from Basilisk to Xode**

Prepare a new open channel request from Basilisk to Xode using the hrmpInitOpenChannel function under the hrmp module in the Kusama relay chain.

Extrinsic: hrmp

Function: hrmpInitOpenChannel

Parameters:

* recipient: 3344 — Parachain ID of Xode.
* proposedMaxCapacity: 1000
* proposedMaxMessageSize: 102400

Once executed, the encoded hash **0x3c00100d00006400000000900100** will be generated, which needs to be saved for use in Step 3.4.

**Step 3.4 Prepare an XCM Message for the Open Channel Request from Basilisk to Xode**

Create an XCM message using the send function under the polkadotXcm module to initiate the open channel request from Basilisk to Xode. The message will include the encoded hash 0x3c00100d00006400000000900100 as a parameter.

This XCM message is prepared but will not be executed until the referenda is approved.

Extrinsic: polkadotXcm

Function: send

Parameters:

* dest (v3):
  + parent: 0
  + interior: Here
* message (v3):
  + WithdrawAsset:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + BuyExecution:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + Transact:
    - originKind: Native
      * refTime: 1000000000
      * proofSize: 65536
    - call:
      * encoded: 0x3c00100d00006400000000900100
  + RefundSurplus
  + DepositAsset:
    - assets: Wild
      * Wild: AllCounted
      * AllCounted: 1
    - beneficiary:
      * parent: 0
      * interior: (X2, (AccountId32(5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY)))

**Step 3.5: Prepare to Register the XON Token on Basilisk**

Prepare the extrinsic to register the XON token on Basilisk by using the register function under the assetRegistry module. This ensures that the XON token will be recognized as an asset on Basilisk.

This extrinsic is part of the batch call in the pre-image but will be executed only after the referenda is approved.

Extrinsic: assetRegistry

Function: register

Parameters:

* name: Xode Native Token
* assetType: Token
* existentialDeposit: 100786131828
* Metadata:
  + symbol: XON
  + decimals: 12
* Location:
  + parent: 0
  + interior: (X2, (Parachain(2034), GeneralIndex(0)))

**Step 4: Create an Encoded Hash in the Kusama Relay Chain**

Once the referenda has been approved, proceed with sending an open channel request from Basilisk to the Kusama relay chain. Use the hrmpAcceptOpenChannel function under the hrmp module in Kusama.

Extrinsic: hrmp

Function: hrmpAcceptOpenChannel

Parameters:

* sender: 2090 — Parachain ID of Basilisk

The encoded hash generated will be 0x3c012a080000, which will be used for Step 5.

**Step 5: Execute the Encoded Hash on Xode Parachain**

Prepare to execute the encoded hash 0x3c012a080000 on the Xode parachain to finalize the open channel request from Kusama to Xode. This will complete the onboarding process of the XON token to the Basilisk parachain, ensuring integration within the Kusama ecosystem.

Use the send function under the polkadotXcm module to prepare the message, which will be executed once the referenda is approved.

Extrinsic: polkadotXcm

Function: send

Parameters:

* dest (v3):
  + parent: 0
  + interior: Here
* message (v3):
  + WithdrawAsset:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + BuyExecution:
    - id: Concrete
      * parent: 0
      * interior: Here
    - fun: Fungible
      * Fungible: 10000000000
  + Transact:
    - originKind: Native
      * refTime: 1000000000
      * proofSize: 65536
    - call:
      * encoded: 0x3c012a080000
  + RefundSurplus
  + DepositAsset:
    - assets: Wild
      * Wild: AllCounted
      * AllCounted: 1
    - beneficiary:
      * parent: 0
      * interior: (X2, (AccountId32(5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY)))

By following these steps, the XON token will be successfully onboarded to the Basilisk parachain, facilitating smooth integration and interoperability within the Kusama ecosystem.

**References:**

* <https://docs.hydration.net/devs/xchain> - Hydration Docs
* <https://docs.bsx.fi/build_xc_integration/> - Basilisk Docs
* <https://docs.polkadot.com/tutorials/interoperability/xcm-channels/para-to-para/> - Polkadot HRMP
* <https://basilisk.subsquare.io/democracy/referenda/36> - Sample Calls
* <https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Frpc.ibp.network%2Fkusama#/explorer> - Kusama Relay Chain
* <https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Frpc.basilisk.cloud#/explorer> - Basilisk Parachain - 2090
* <https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Frpcnodea01.xode.net%2Fn7yoxCmcIrCF6VziCcDmYTwL8R03a%2Frpc#/explorer> - Xode Parachain - 3344