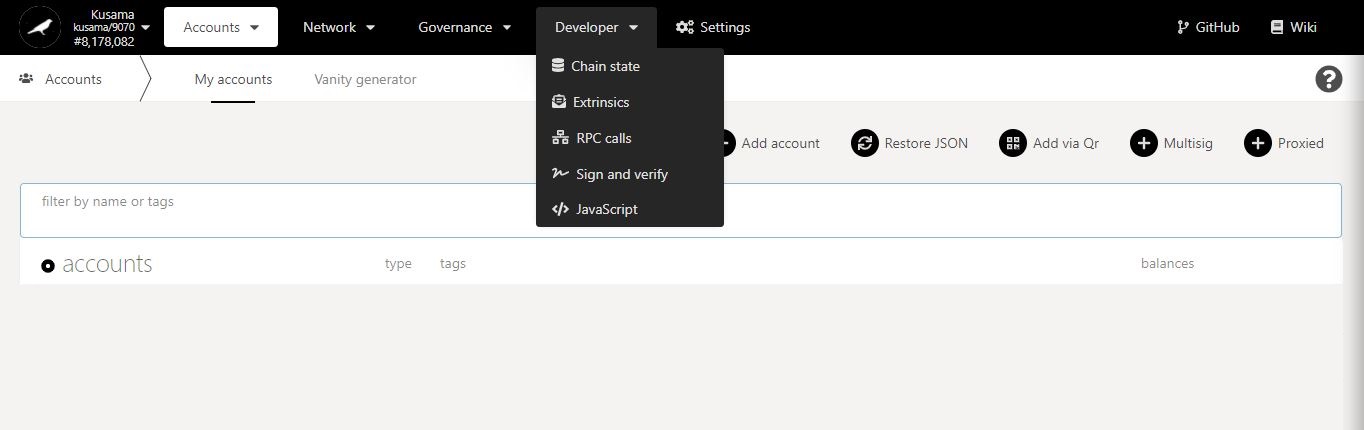
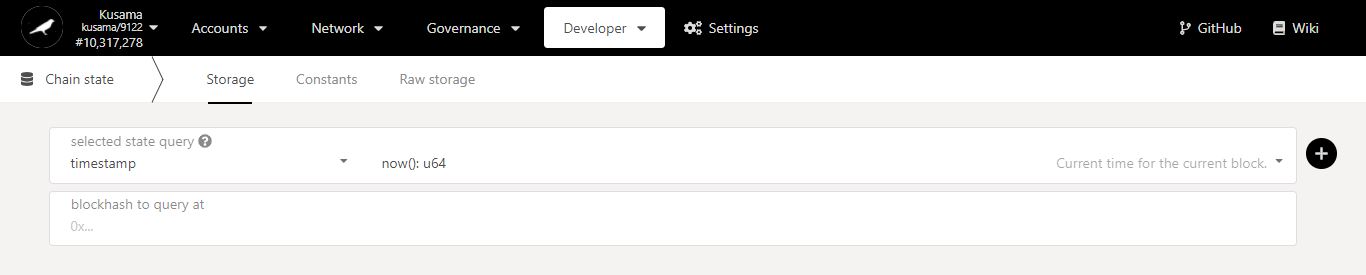
**PART VIII: Developer**

1. **Chain state: Make API calls to query on-chain data.**

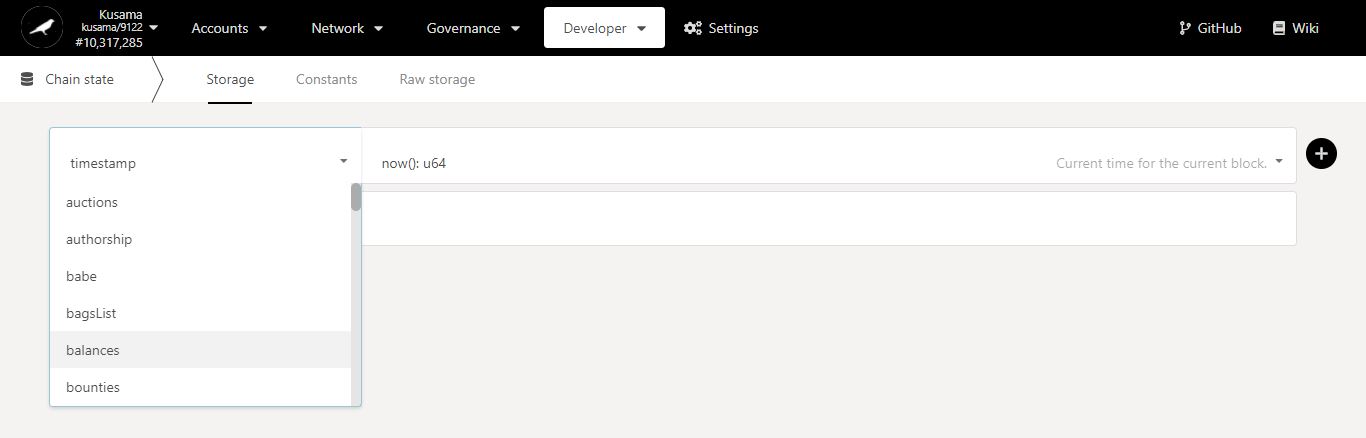


* 1. **View storage items.**

1. Click **Storage**. **account**.

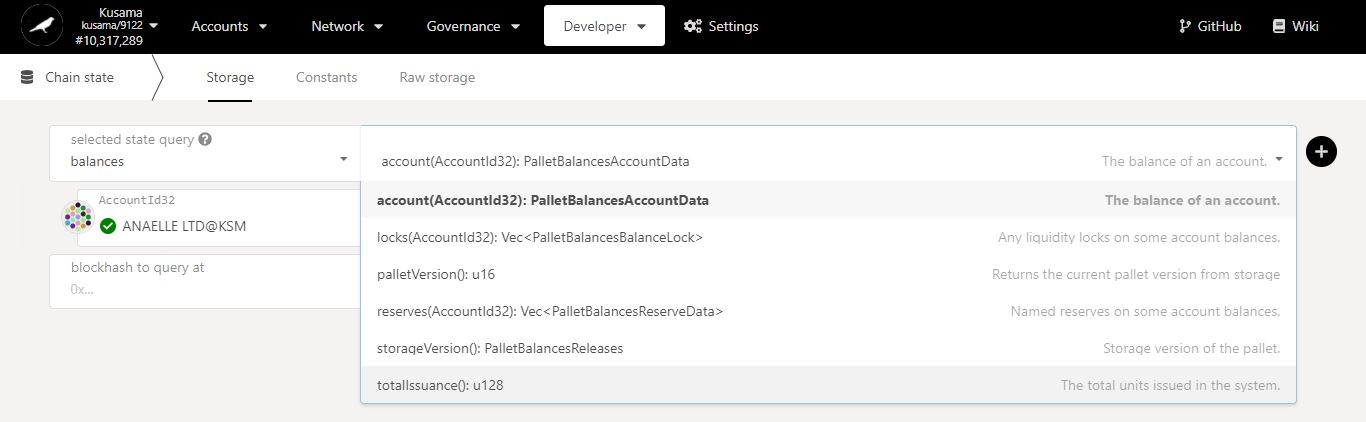


Interface for **accessing information about the state of the chain** stored on-chain.



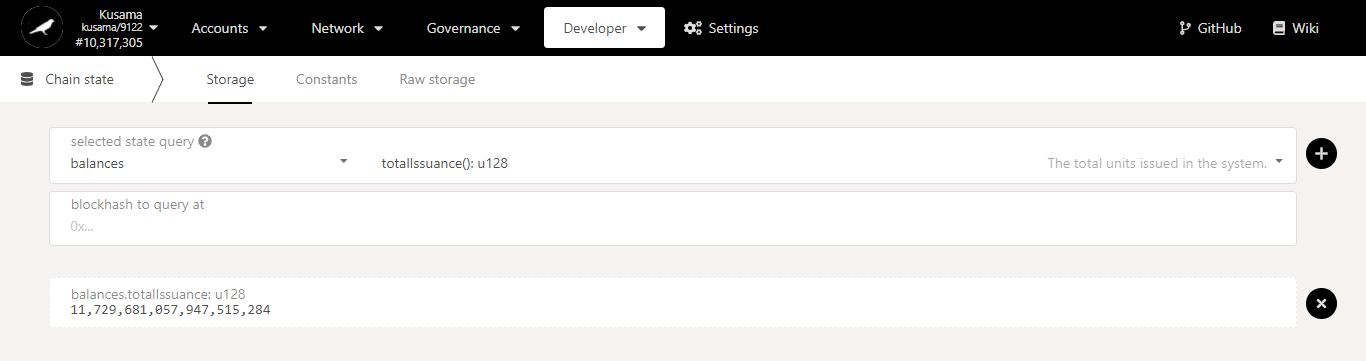
2. Click on the **dropdown arrow** to view a list of storage items to query.

3. Select one **storage item**.



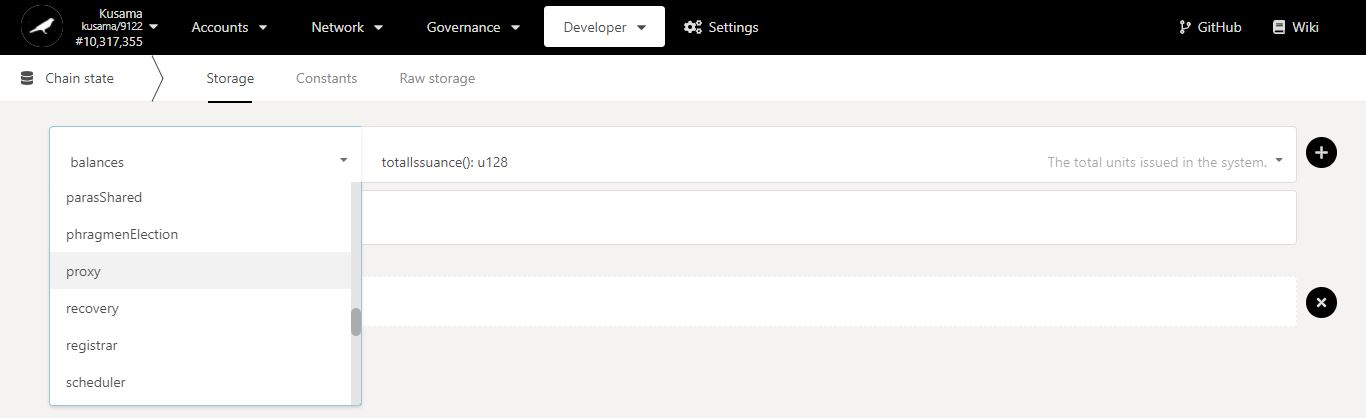
5. Select one **method**.

4. Click on the **dropdown arrow** to view a list of methods for building queries.

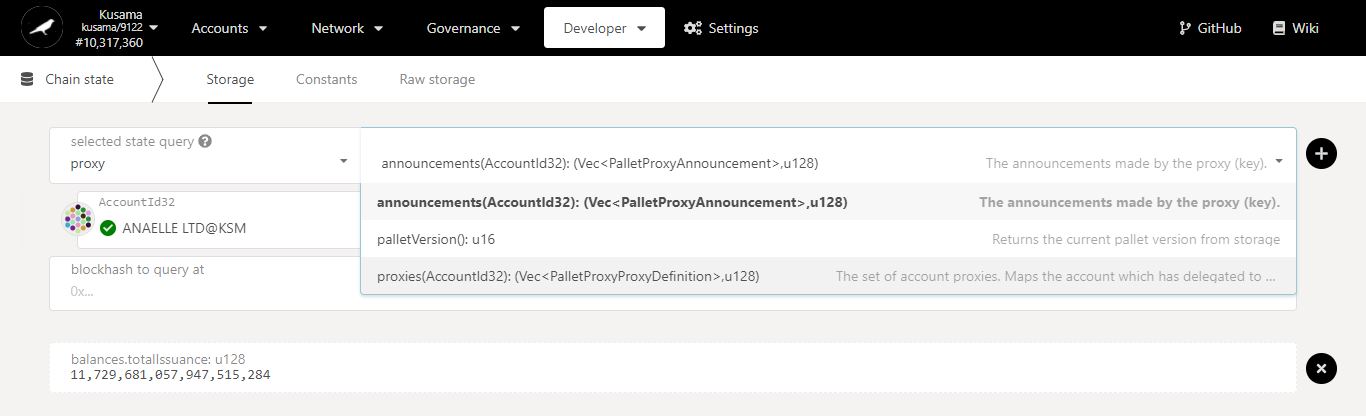


**7. The information requested is now available on the interface!**

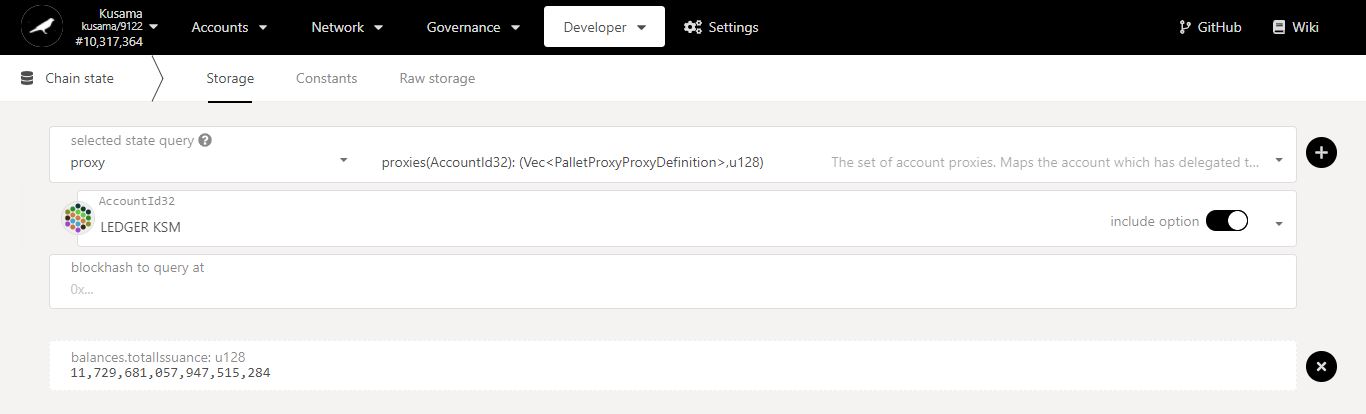
6. Click on the **+** button to submit your query.



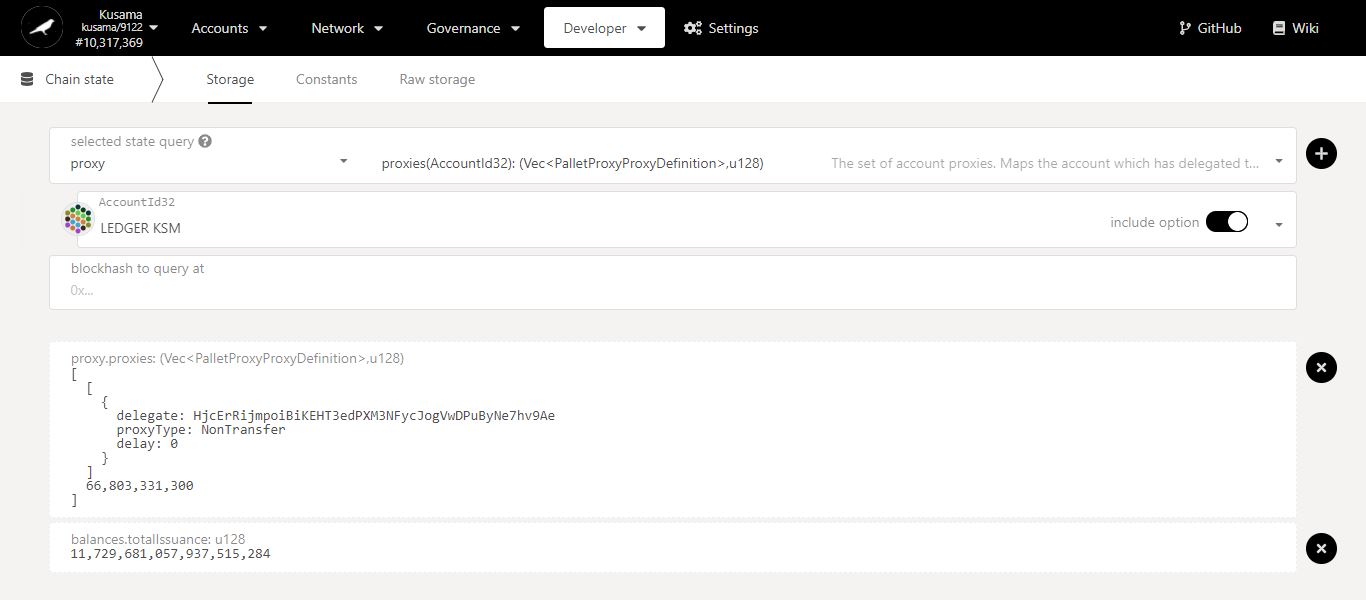
8. You can continue to build different queries by selecting another **storage item…**



9. …and another **method**…



10. …then all the necessary **options**.

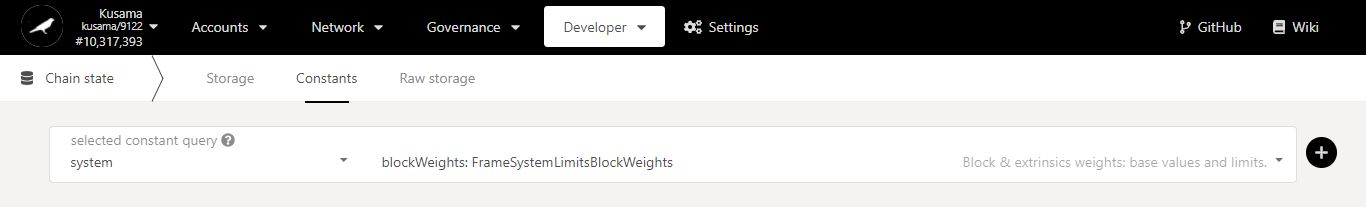


**12. The new information requested is now stacked on the interface!**

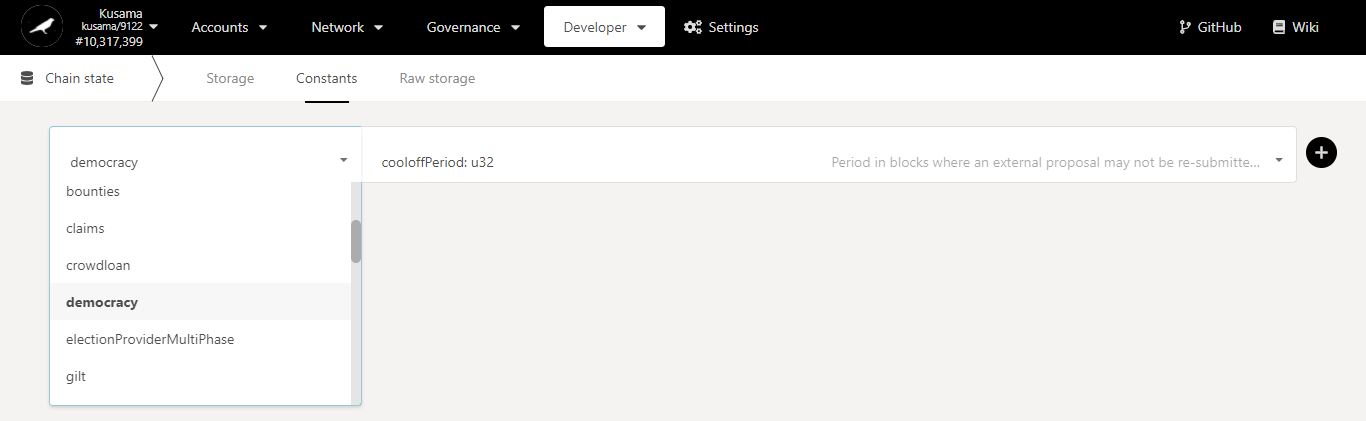
11. Click on the **+** button to submit your second query.

* 1. **View chain constants.**

1. Click **Constants.**

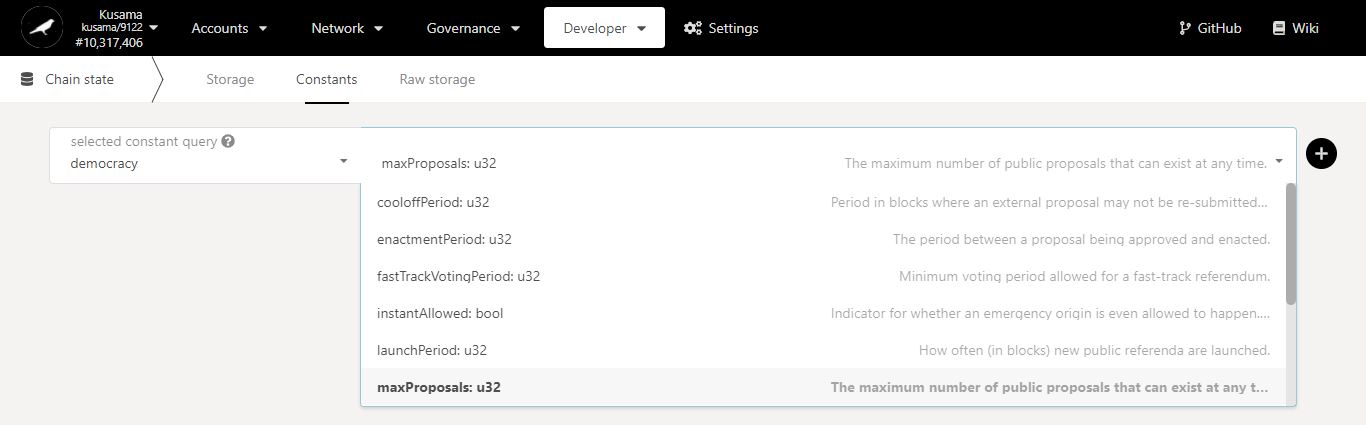


Interface for **accessing information about predefined chain parameters** stored on-chain.



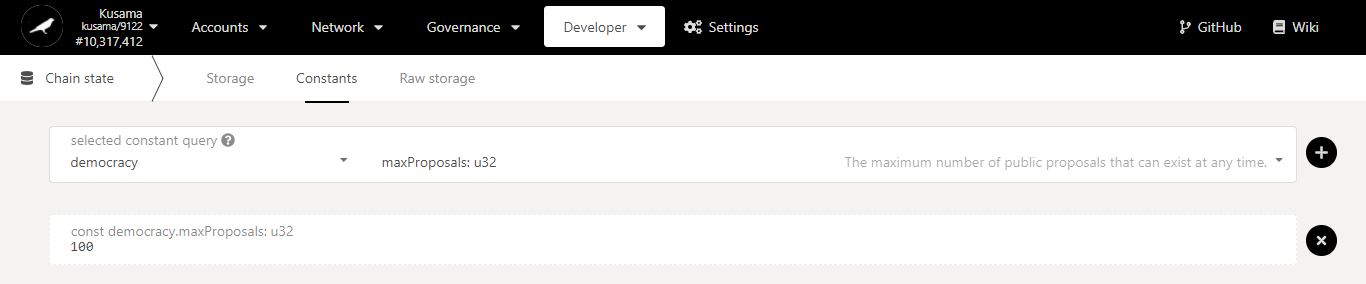
3. Select one **chain constant**.

2. Click on the **dropdown arrow** to view a list of chain constants to query.



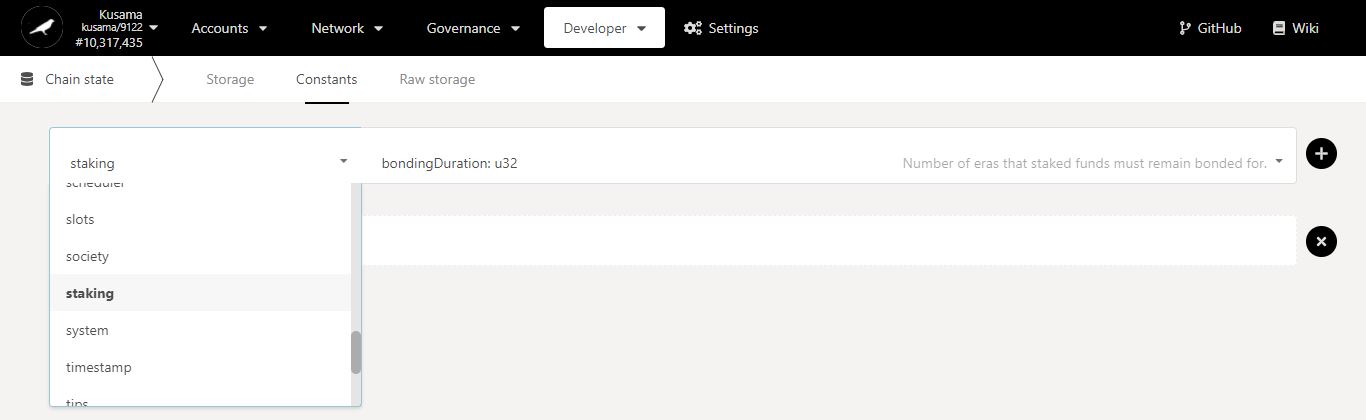
5. Select one **method**.

4. Click on the **dropdown arrow** to view a list of methods for building queries.

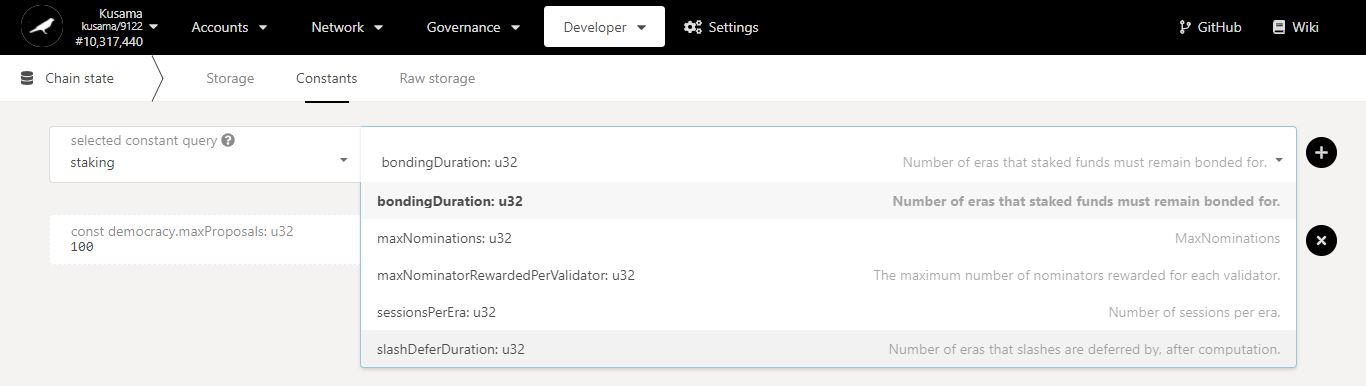


6. Click on the **+** button to submit your query.

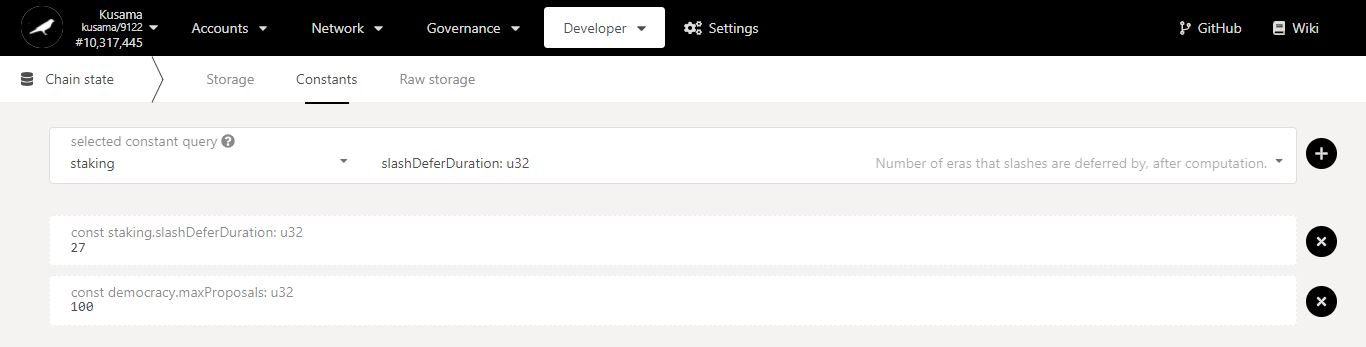
**7. The information requested is now available on the interface!**



8. You can continue to build different queries by selecting another **chain constant…**



9. …and another **method**…

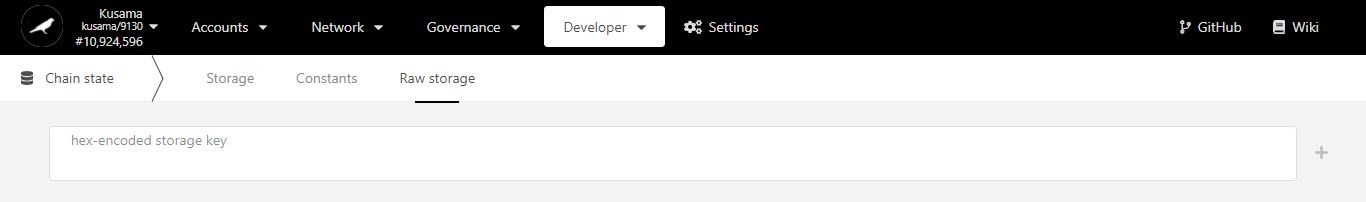


**11. The new information requested is now stacked on the interface!**

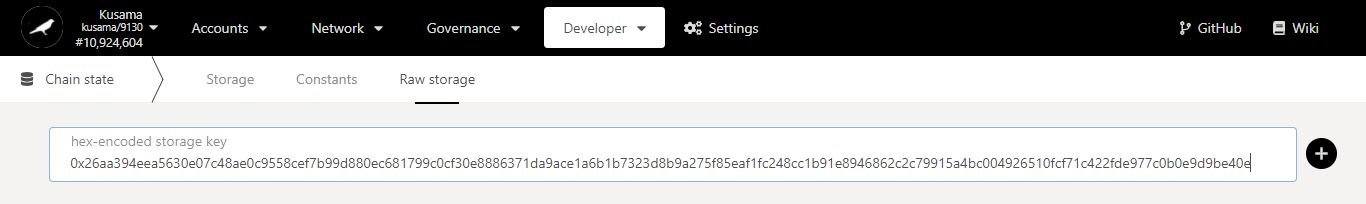
10. Click on the **+** button to submit your second query.

* 1. **Retrieve raw storage information.**

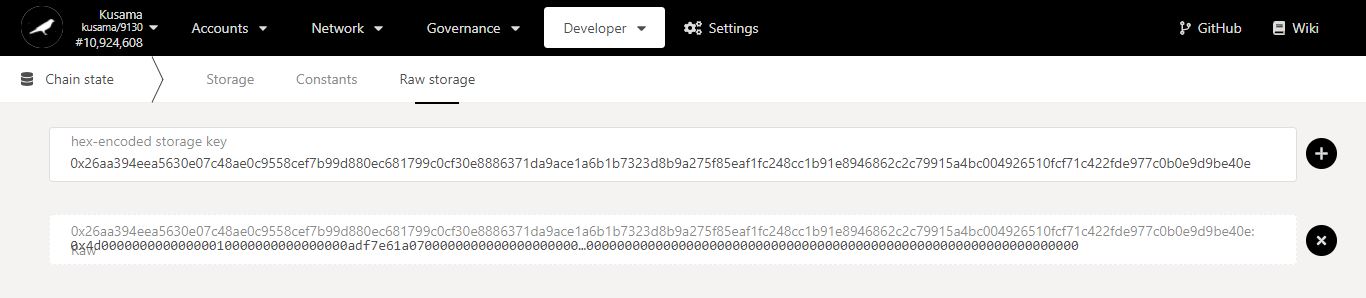
1. Click **Raw storage.**



Interface for **accessing information about storage maps** stored on-chain.



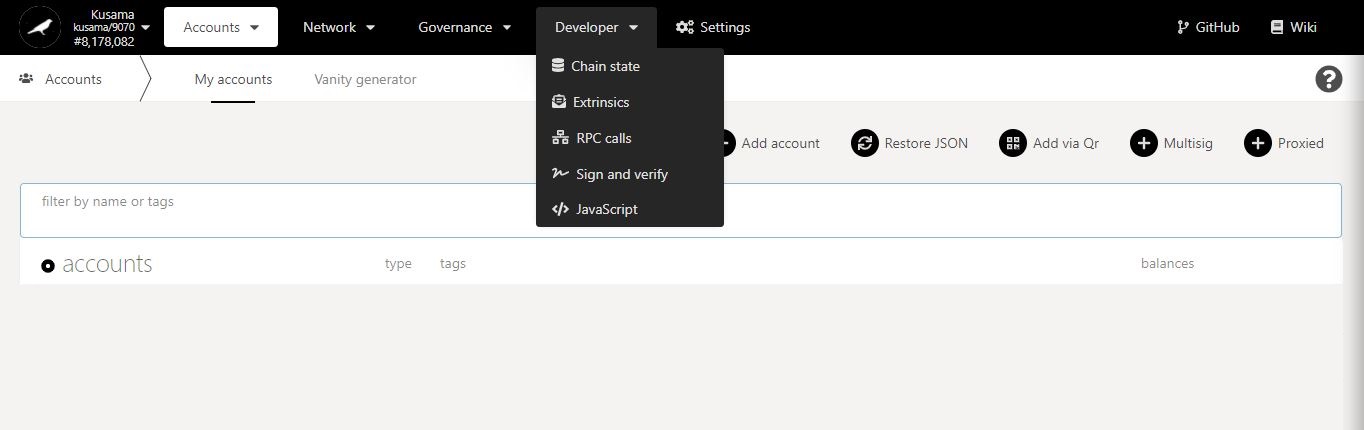
2. Provide the **storage key** constructed for the map to query and click the **+** button.



You can find more about Storage keys construction and Storage maps on this page: <https://www.shawntabrizi.com/substrate/transparent-keys-in-substrate/>

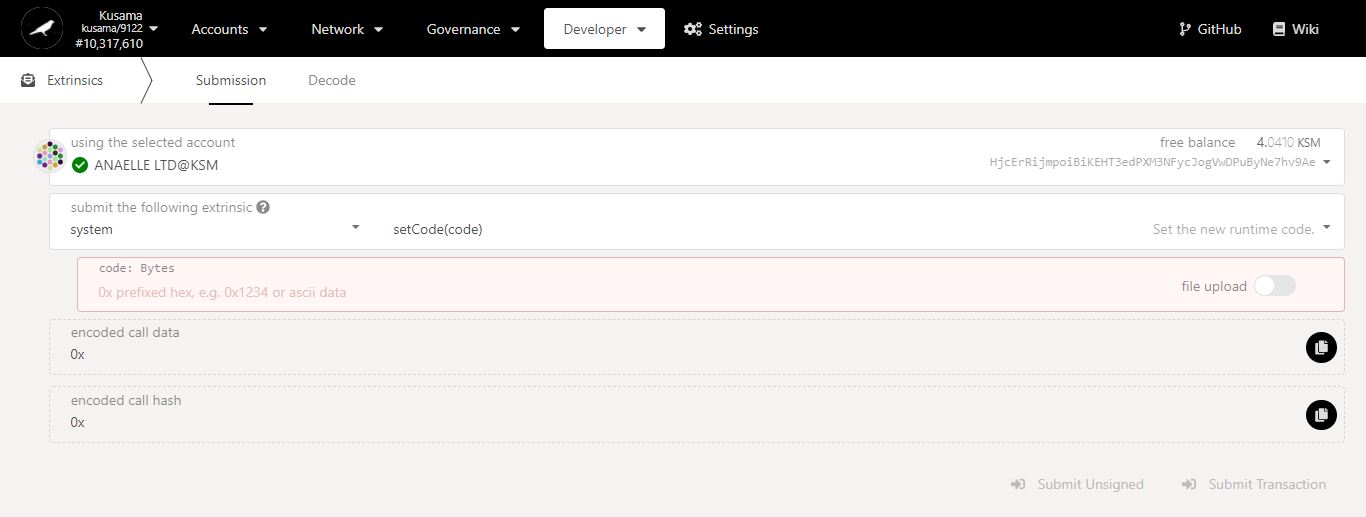
**3. The information requested is now available on the interface, ready to be further decoded!**

1. **Extrinsics: Make API calls to submit data onto the chain directly.**

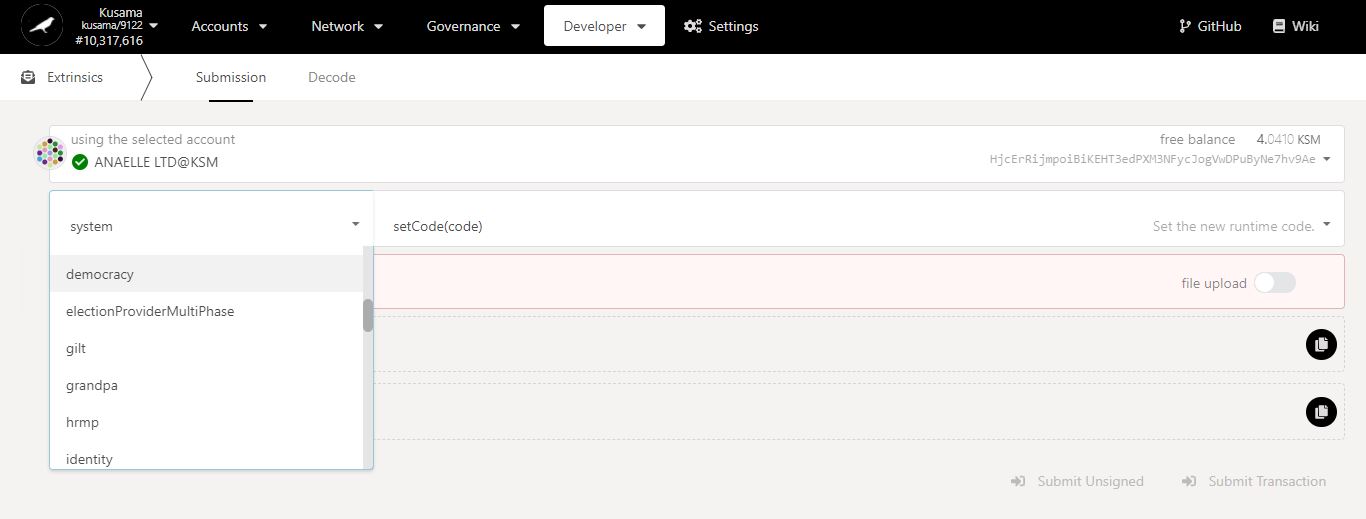


* 1. **View runtime modules and methods.**

1. Click **Submission**. **account**.

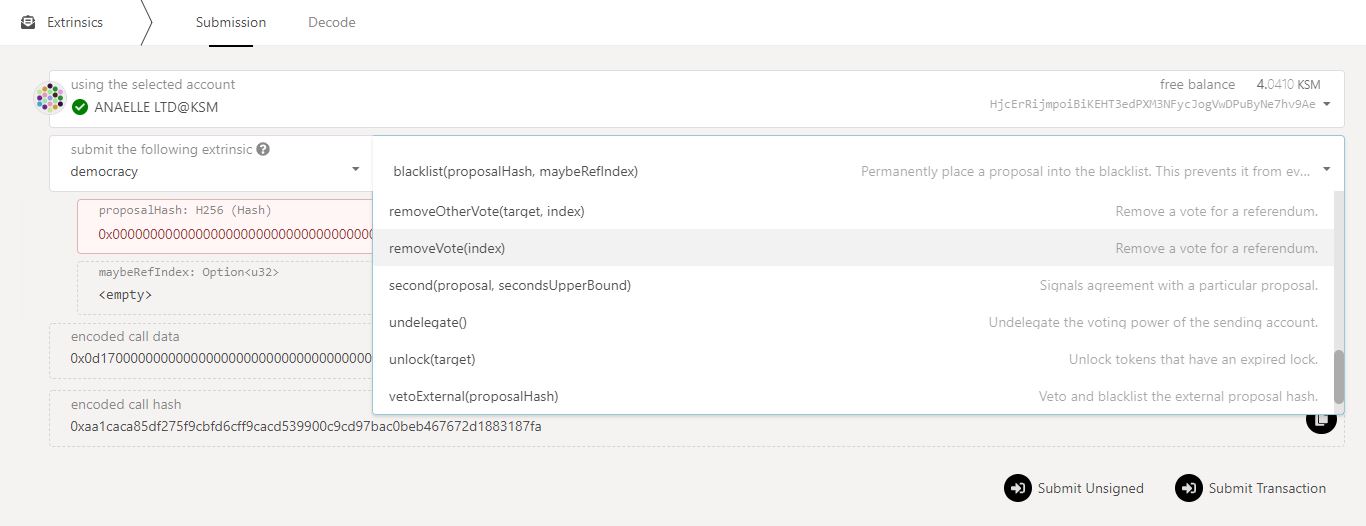


Interface for **submitting external information** onto the chain.



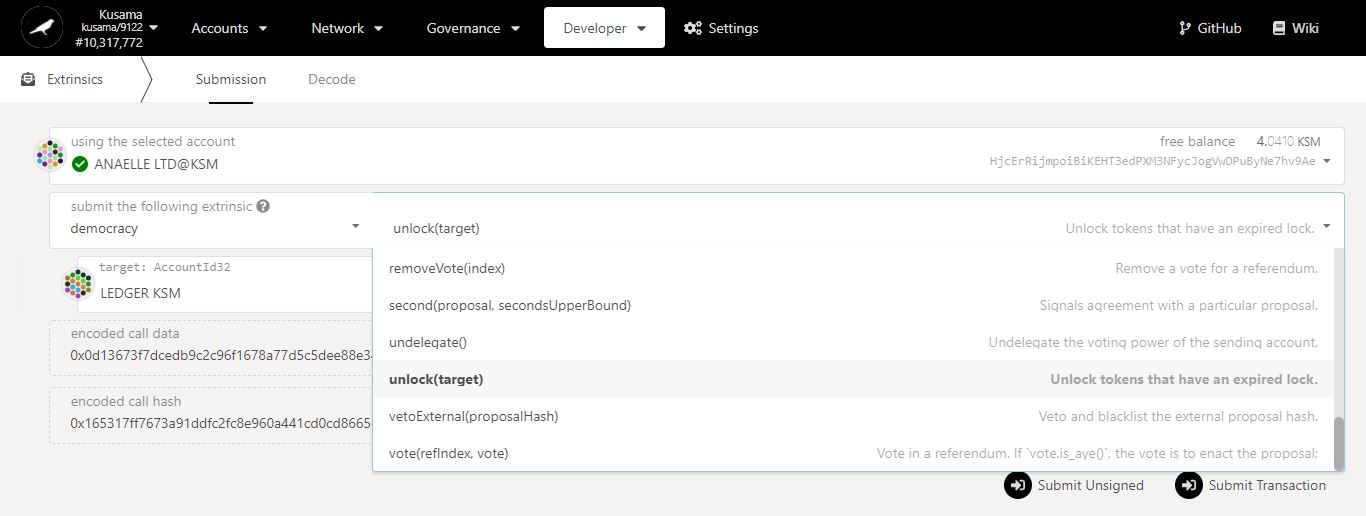
2. Click on the **dropdown arrow** to view all the available runtime modules.

3. Select one **module**.

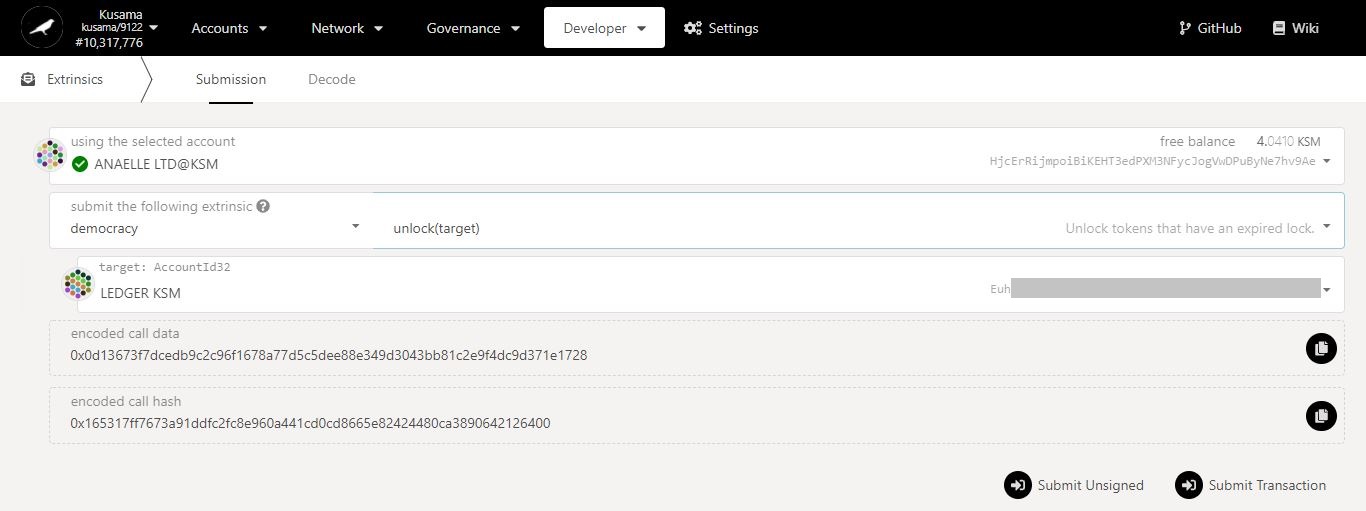


5. **Scroll down** to find the relevant method for your call.

4. Click on the **dropdown arrow** to view all the available methods for this module.



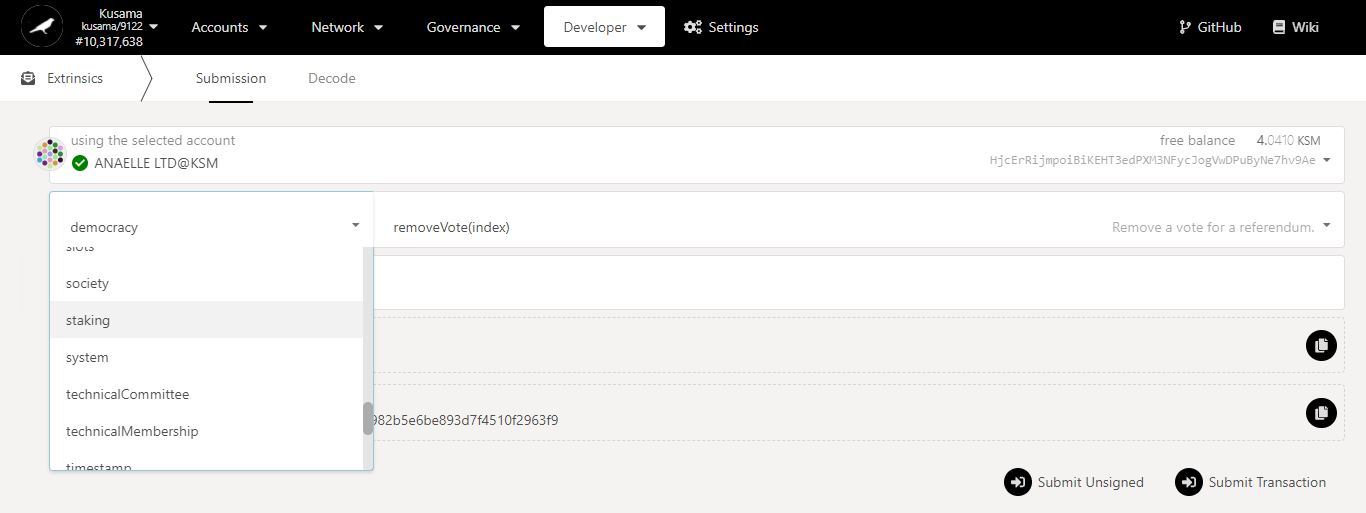
6. Select one **method**.



**7. The module and method requested for the call are now available on the interface!**

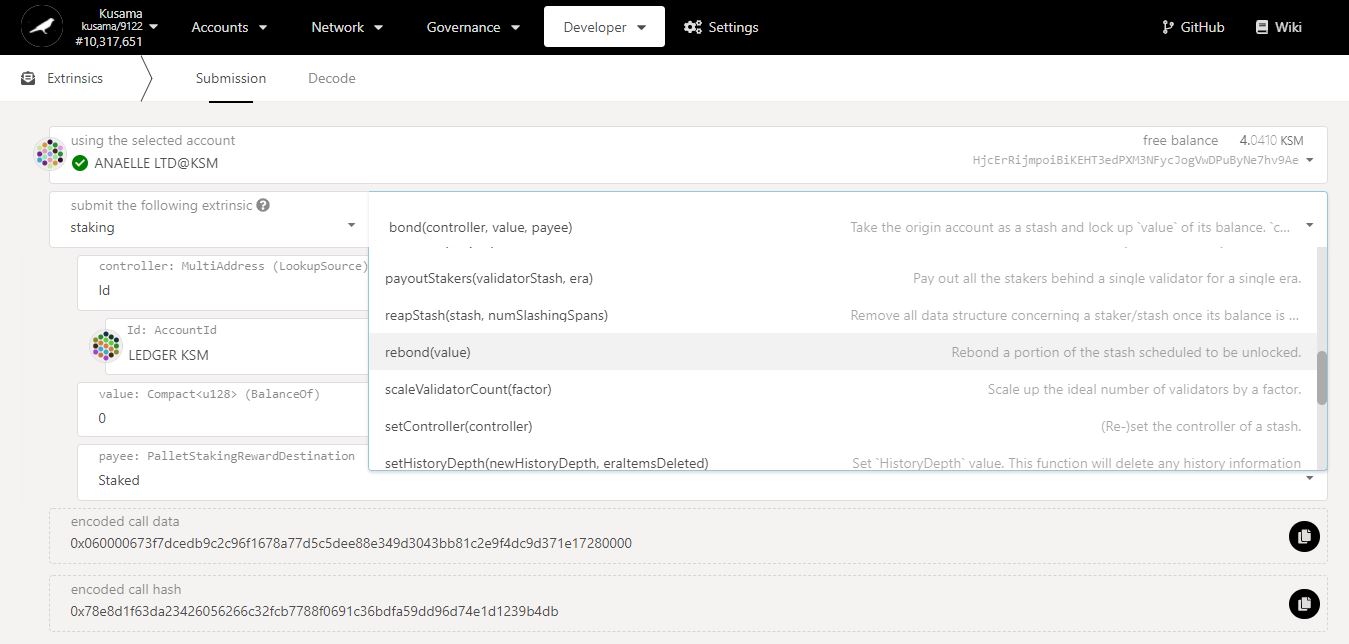
* 1. **Submit calls.**

1. Click **Submission**. **account**.



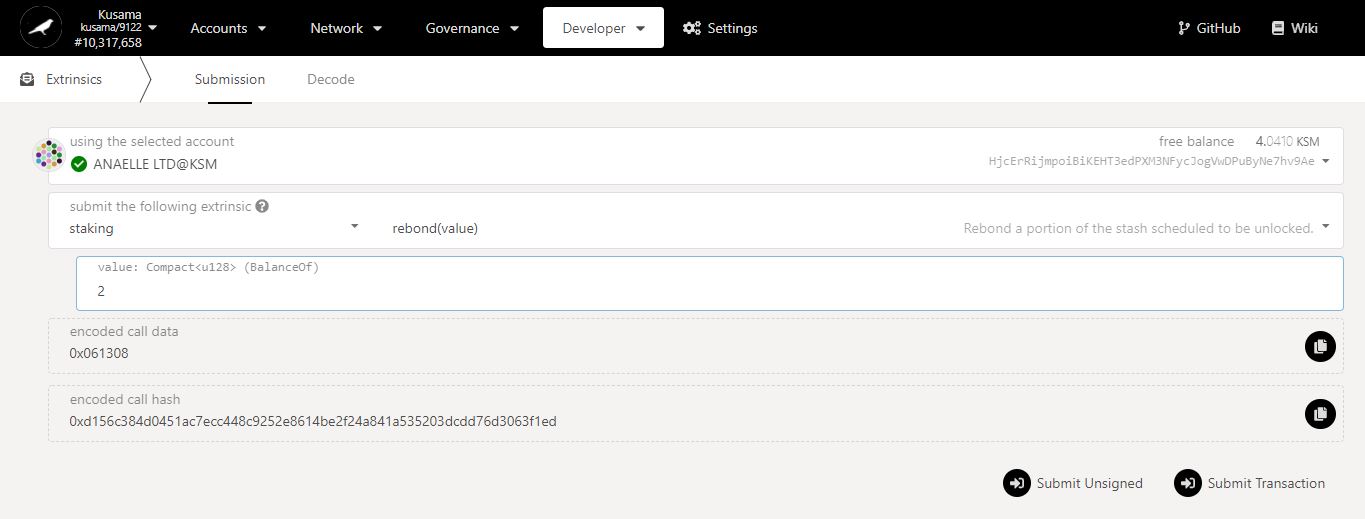
3. Select one **module**.

2. Click on the **dropdown arrow** to view all the available runtime modules.



5. Select one **method**.

4. Click on the **dropdown arrow** to view all the available methods for this module.

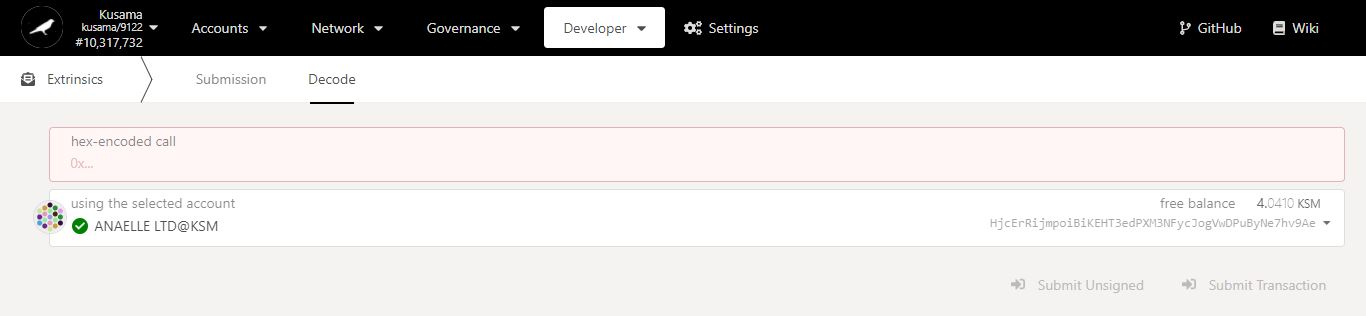


7. Click on **Submit Transaction** to continue the procedure.

6. Enter **the information required**.

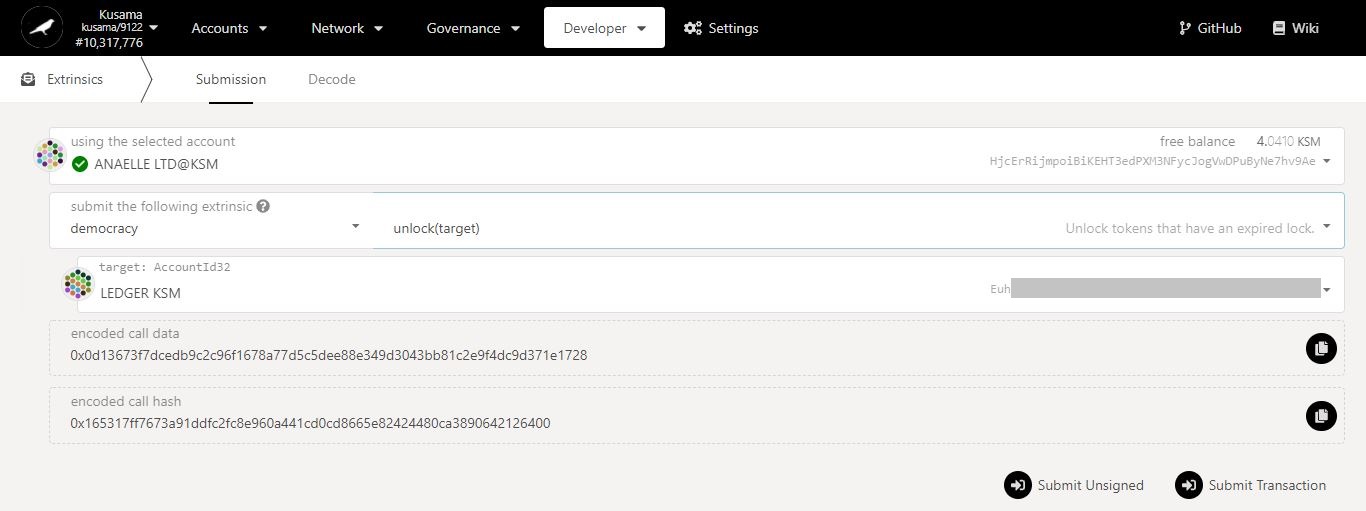
* 1. **Decode hex-encoded calls.**

1. Click **Decode**.



Interface for **decoding external information (to be) submitted** onto the chain.

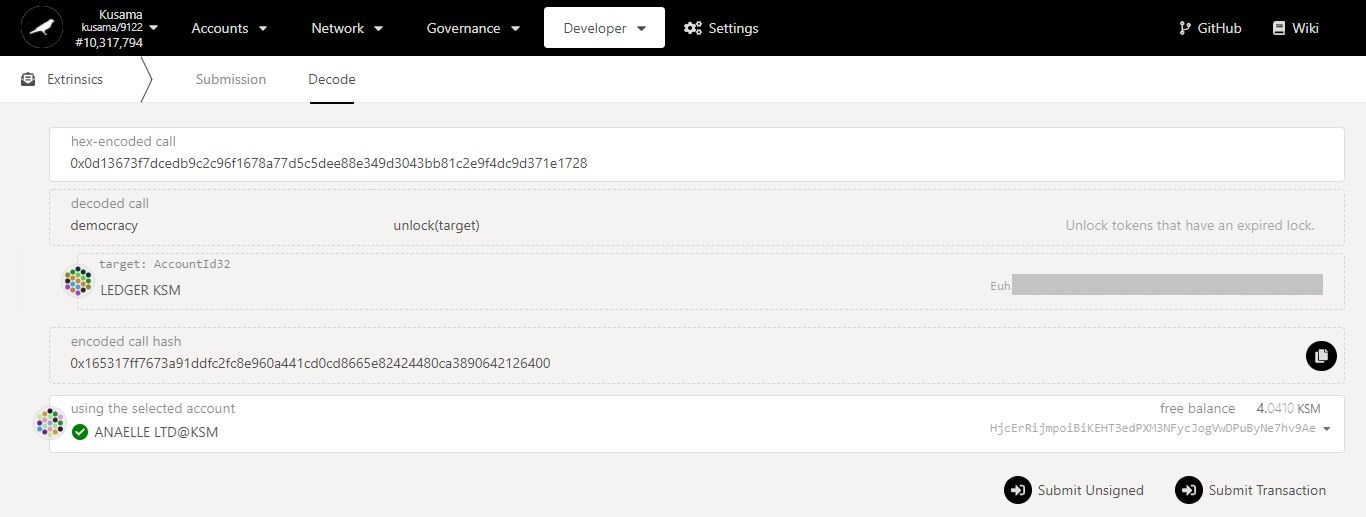
2. Click **Submission**. **account**.



4. Copy **the call data and store it** for future use.

3. Select **the module and method** required for your call.

5. Click **Decode**.

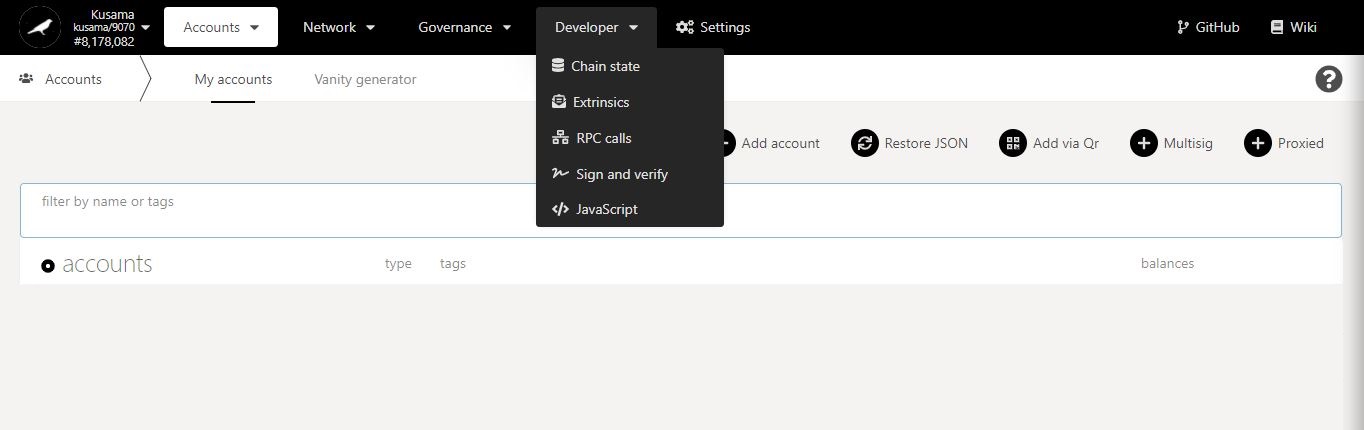


**7. The module and method of the call are now loaded on the interface!**

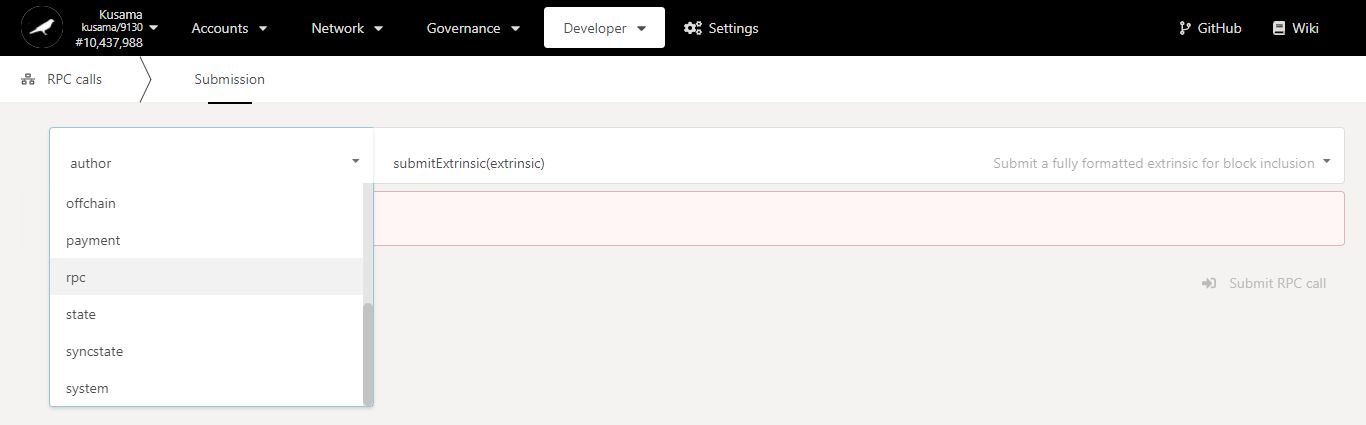
6. Paste **the call data previously stored.**

8. Click on **Submit Transaction** to continue the procedure.

1. **RPC calls: Make RPC calls to submit data onto the chain remotely.**

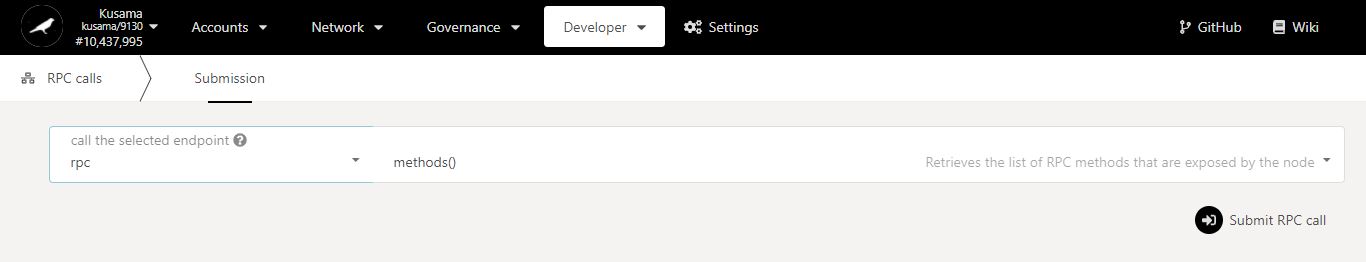


* 1. **View JSON RPC modules and methods.**



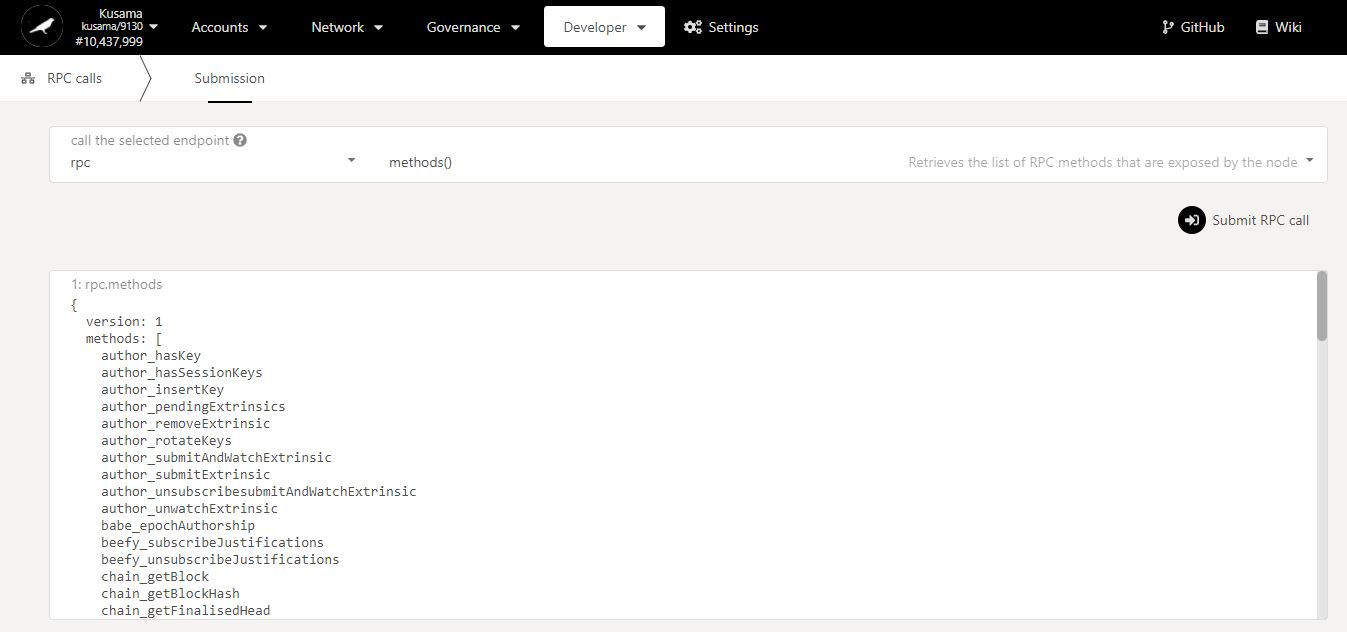
2. Select **RPC**.

1. Click on the **dropdown arrow** to view all the available RPC modules.



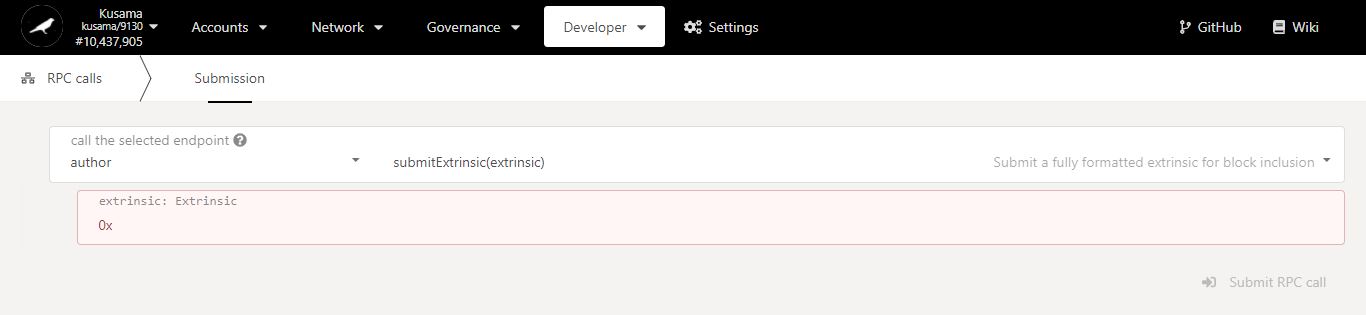
4. Click on **Submit RPC call** to continue the procedure.

3. The single **method available for the RPC module** is loaded.

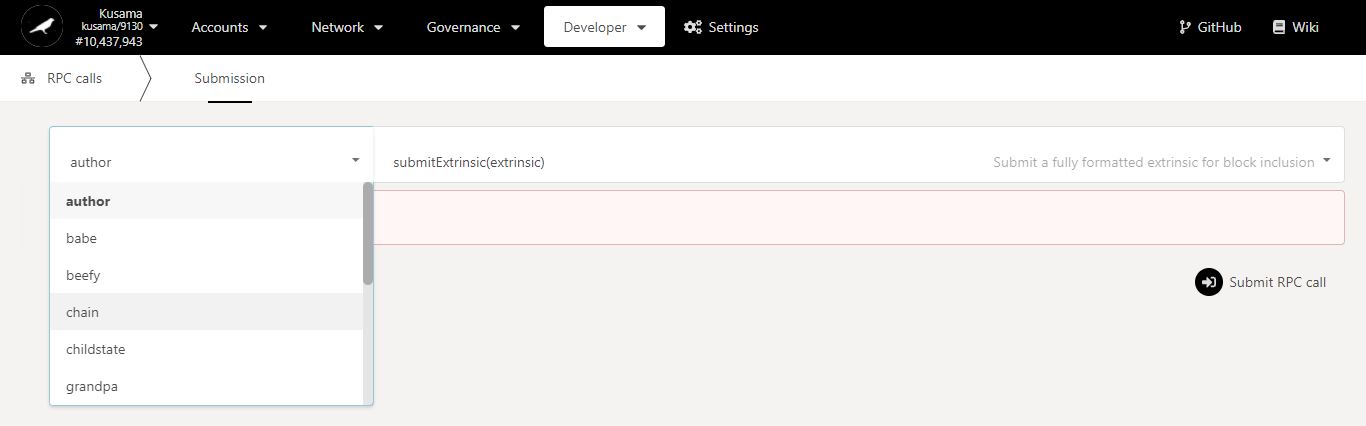


**5. All RPC modules and methods available are now listed on the interface, ready to be used remotely!**

* 1. **Submit RPC calls.**

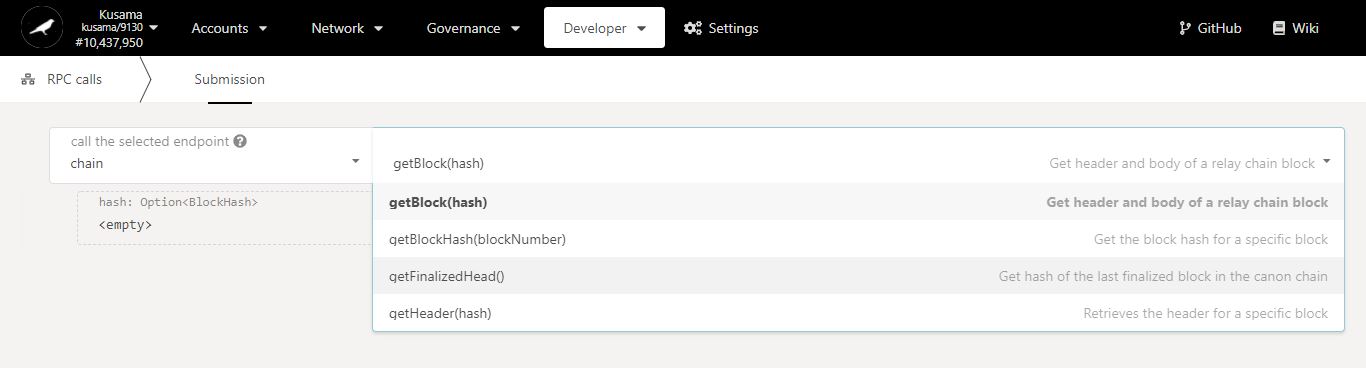


Interface for **submitting external information** onto the chain remotely.



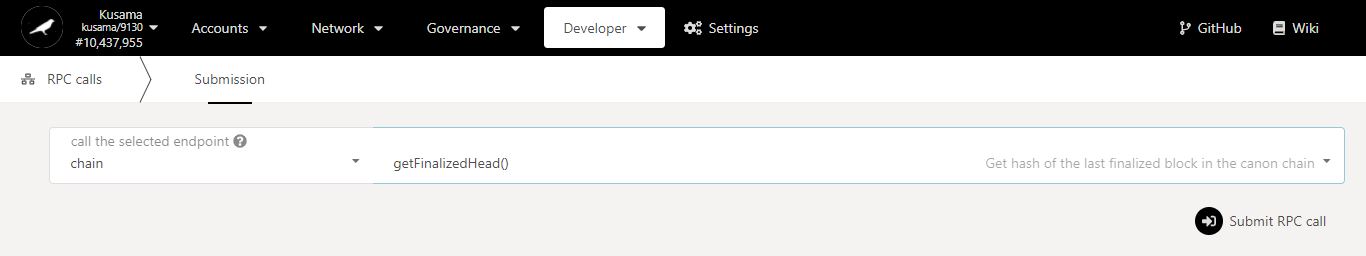
1. Click on the **dropdown arrow** to view all the available RPC modules.

2. Select one **module**.

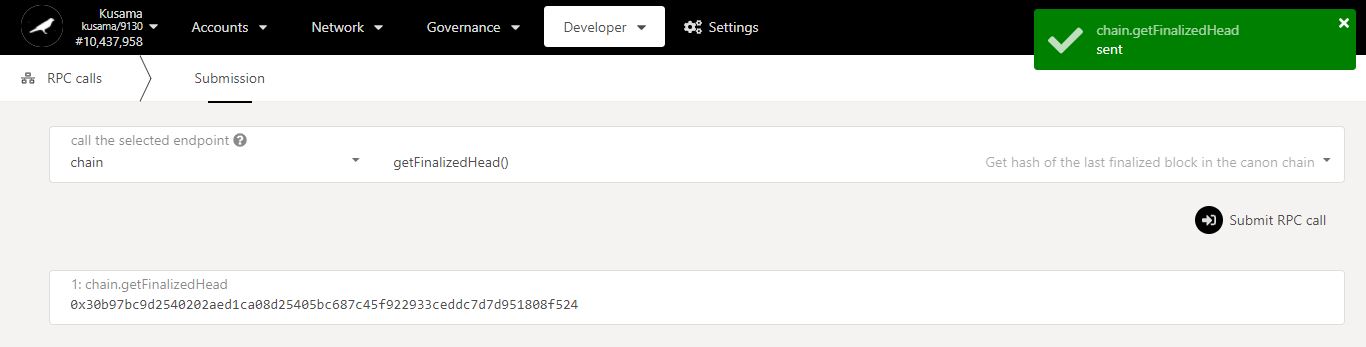


4. Select one **method**.

3. Click on the **dropdown arrow** to view all the available methods for this module.

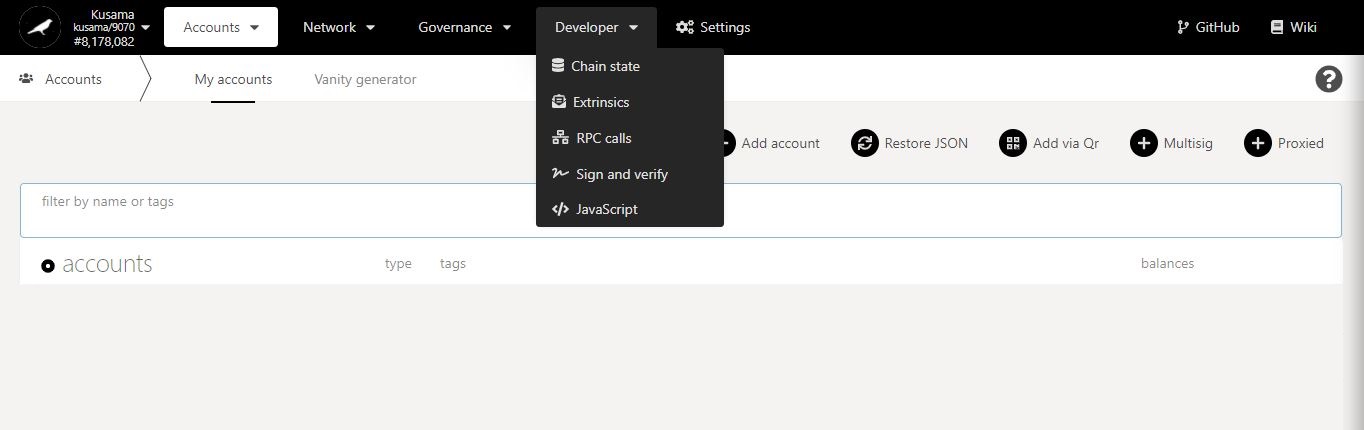


5. Click on **Submit RPC call** to continue the procedure.



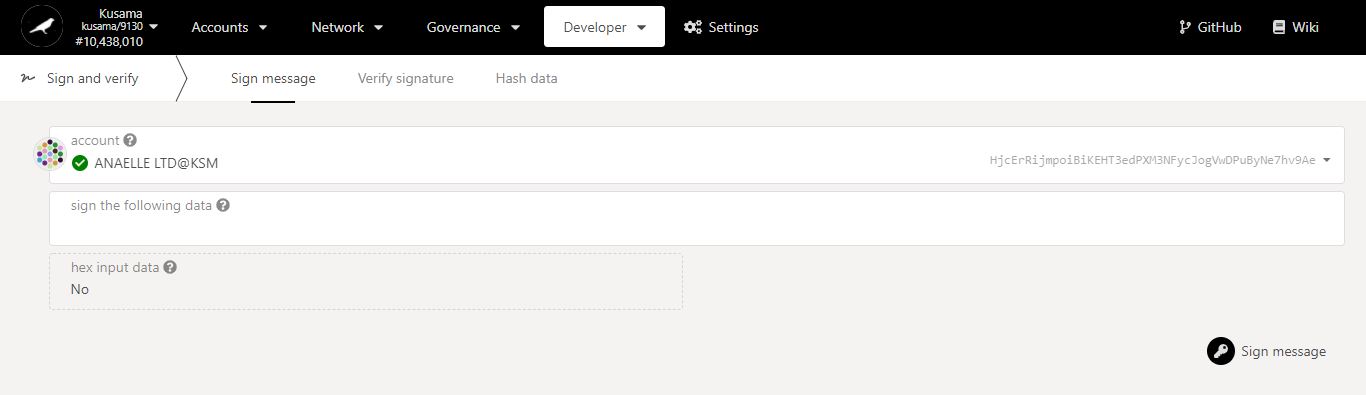
**5. The information submitted has now returned a response on the interface!**

1. **Sign & Verify: Sign messages onto the chain and verify on-chain signatures.**

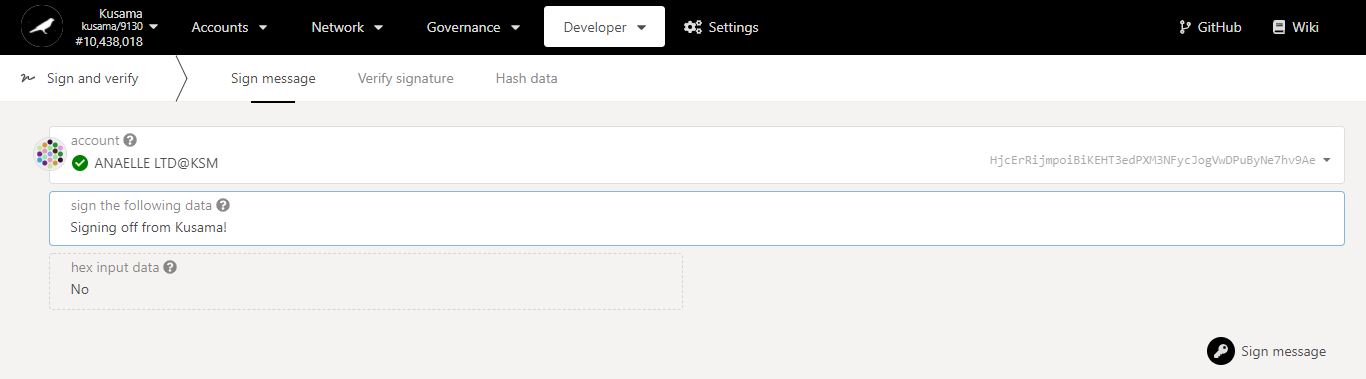


* 1. **Sign messages.**

1. Click **Sign message**.



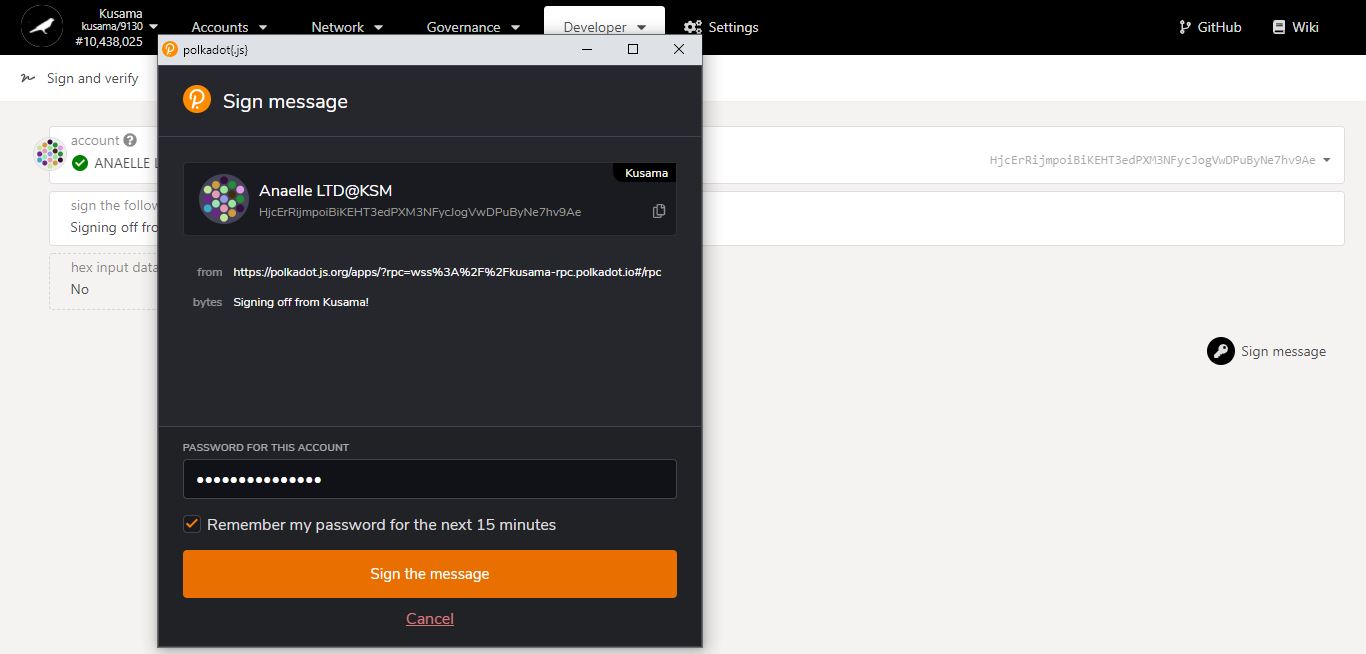
Interface for **signing messages** onto the chain.



4. Click on **Sign message** to continue the procedure.

2. Select the **account** that will sign the message.

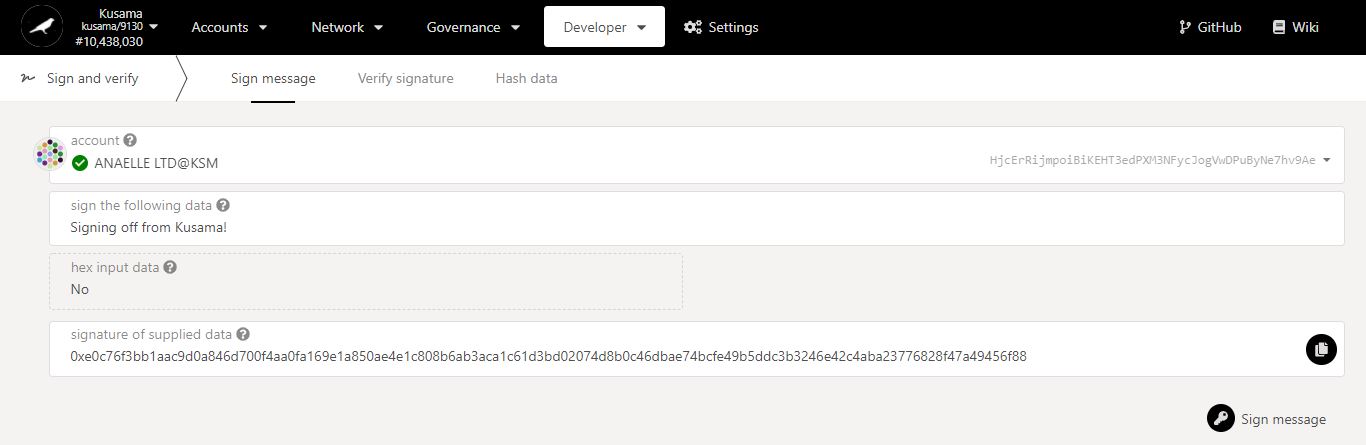
3. Enter the **message** to be signed.



6. Click on **Sign the message** to complete the procedure.

5. Enter **your account’s password** and tick the box to **remember your password, if necessary**.

**Summary** of the transaction sent via the Polkadot-JS extension.

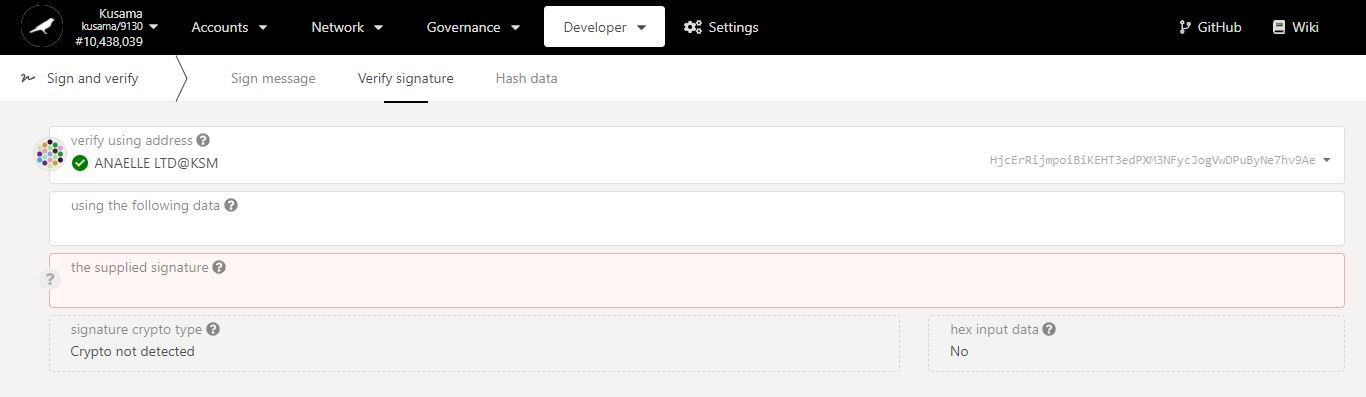


8. Copy the **signature and store it** for future use.

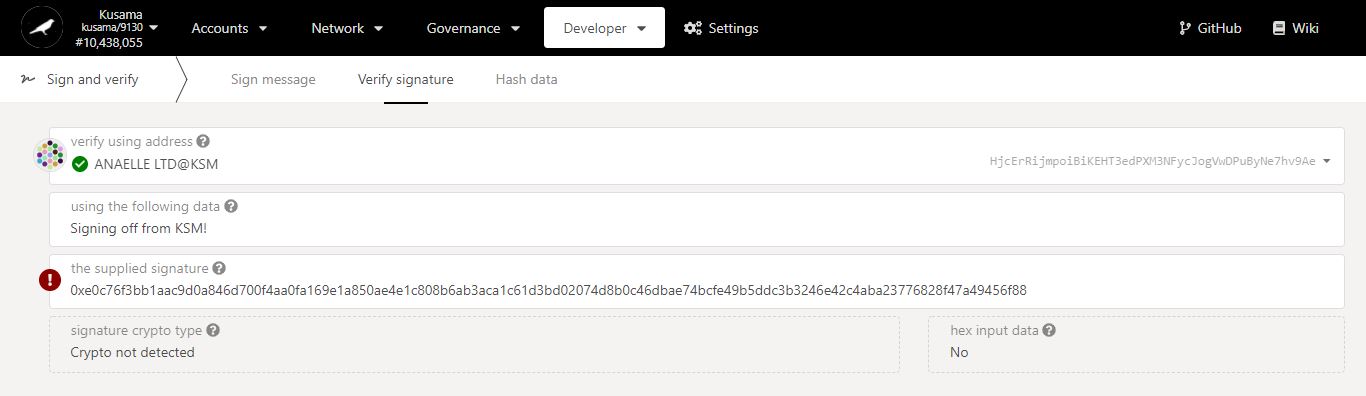
**7. A signature of the signed messaged is now available on the interface!**

* 1. **Verify signatures.**

1. Click **Verify signature**.



Interface for **verifying signatures** stored on-chain.

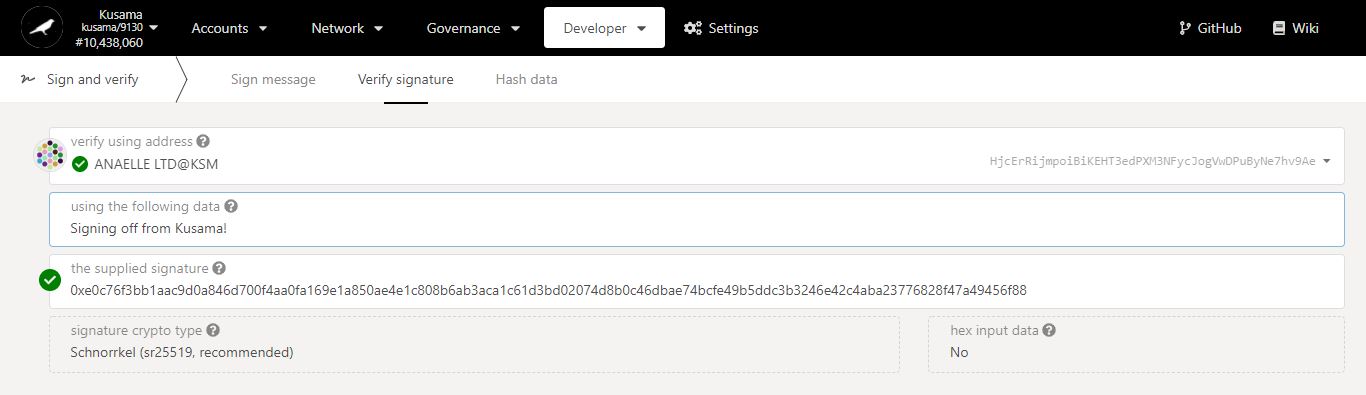


4. Paste the **signature previously stored.**

**5. If any of the information provided is incorrect, the signature will be flagged as invalid!**

3. Enter the **message** that was signed.

2. Select the **account** that signed the message.

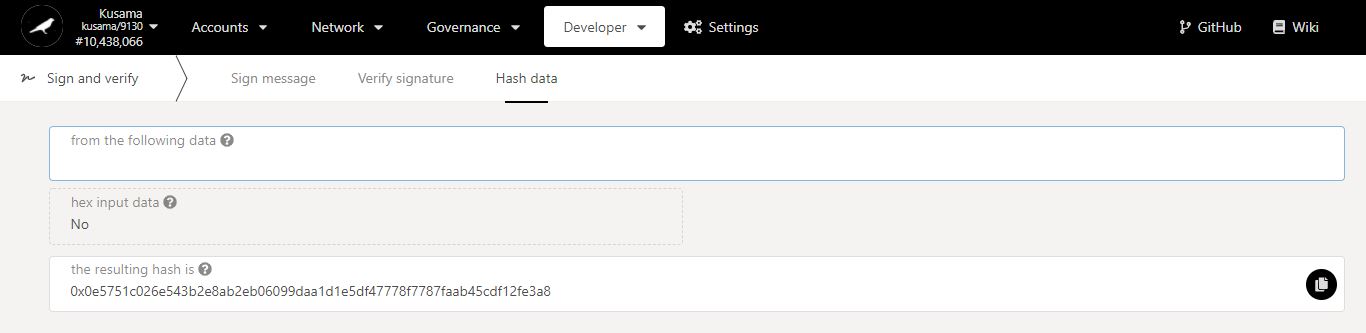


6. **Check and correct** the message that was signed.

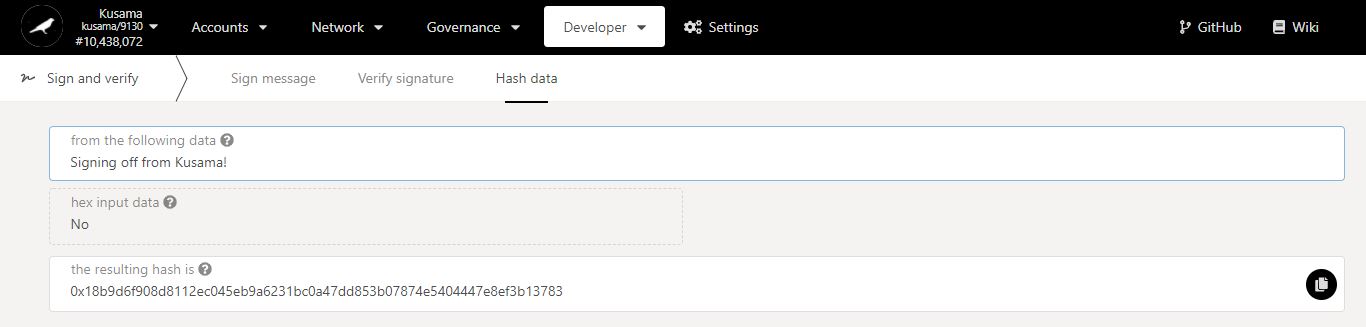
**7. Once all the information provided is correct, the signature will be marked as valid!**

* 1. **Hash data.**

1. Click **Hash data**.



Interface for **encrypting messages** to be signed onto the chain.

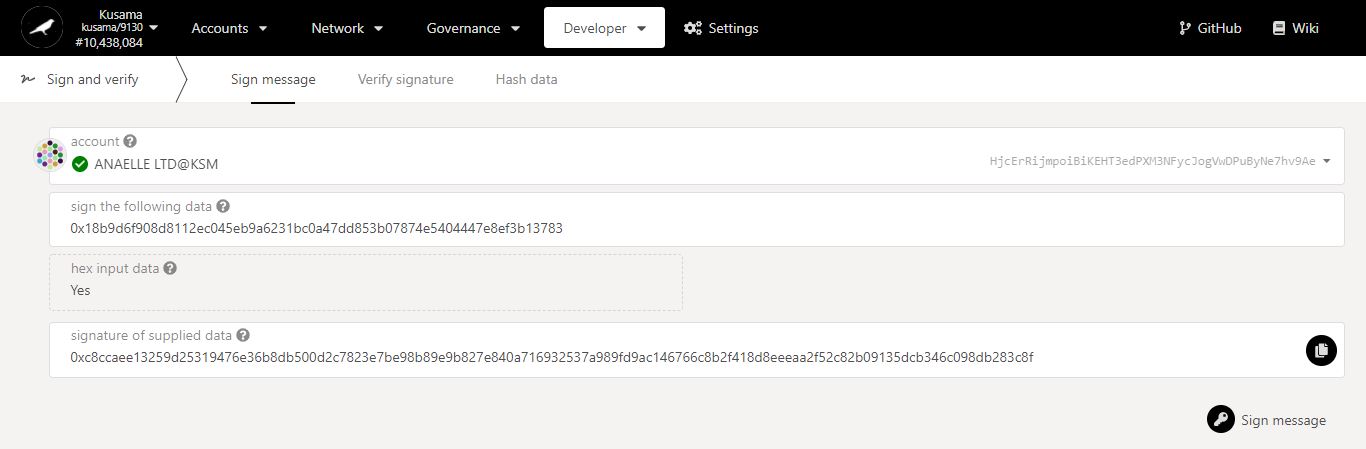


2. Enter the **message** to encrypt.

**3. A hash of the message is now available on the interface!**

4. Copy the **hash and store it** for future use.

5. Click **Sign message**.



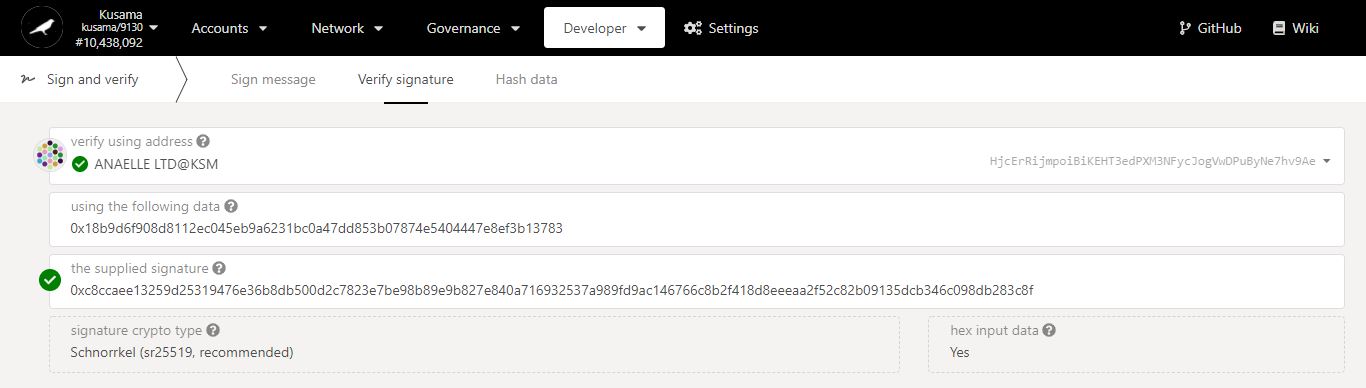
**9. A signature of the hash (encrypted message) will be available on the interface!**

8. Click on **Sign message** to continue the procedure.

7. Paste the **hash previously stored**.

6. Select the **account** that will sign the message.

10. Click **Verify signature**.

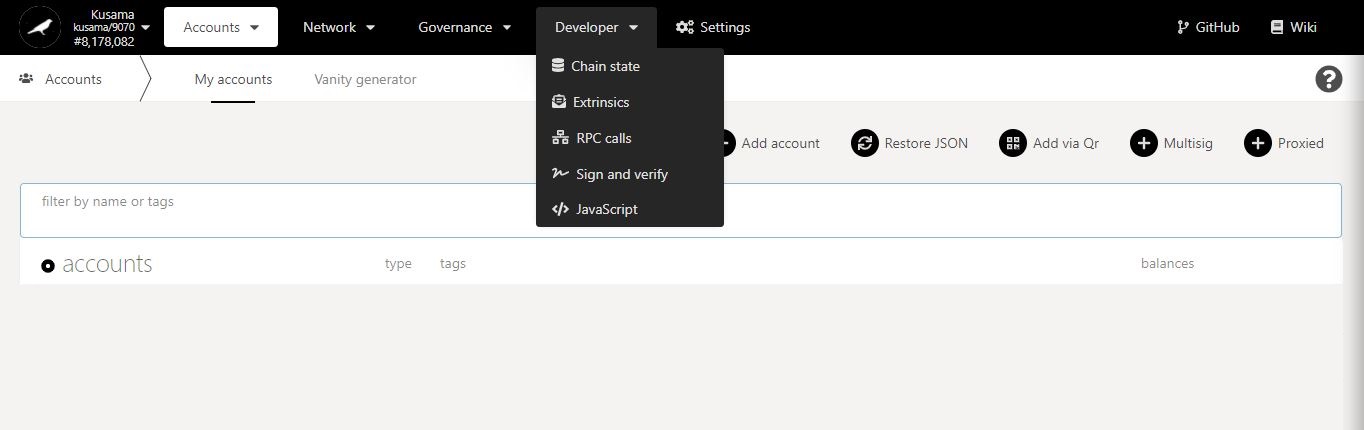


13. Paste the **signature of the hash (encrypted message)** for verification.

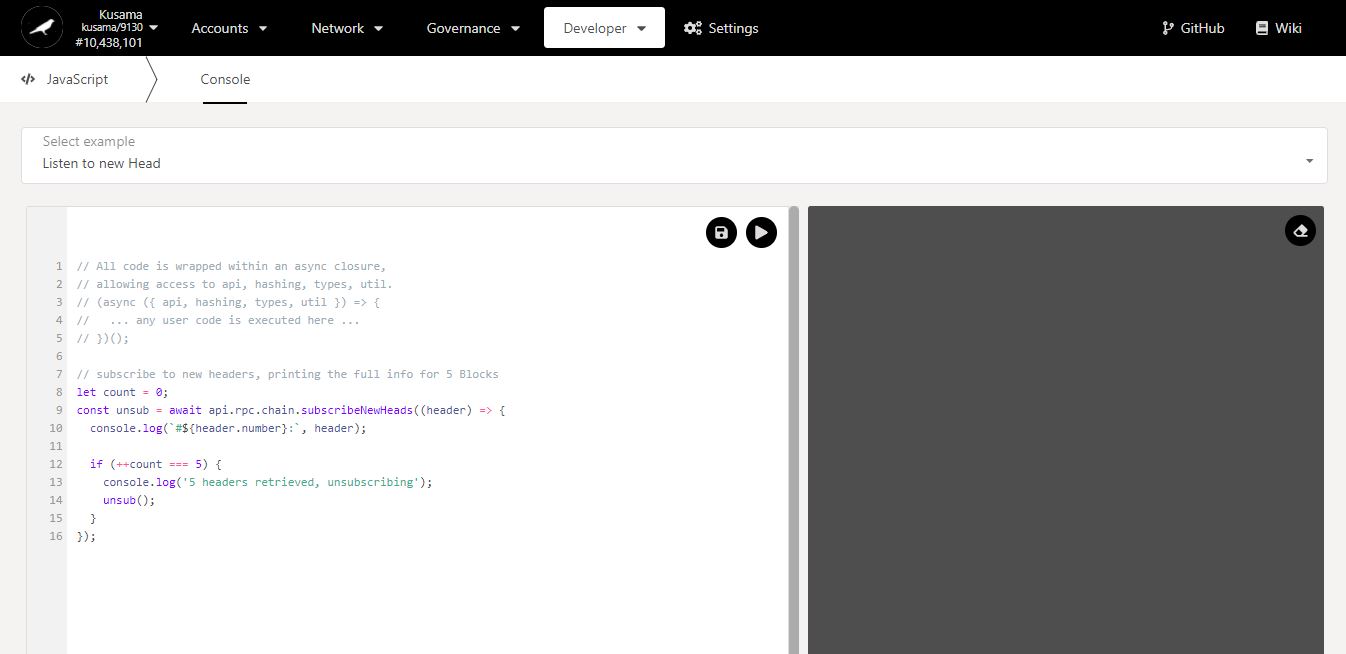
12. Enter the **hash (encrypted message)** that was signed.

11. Select the **account** that signed the encrypted message.

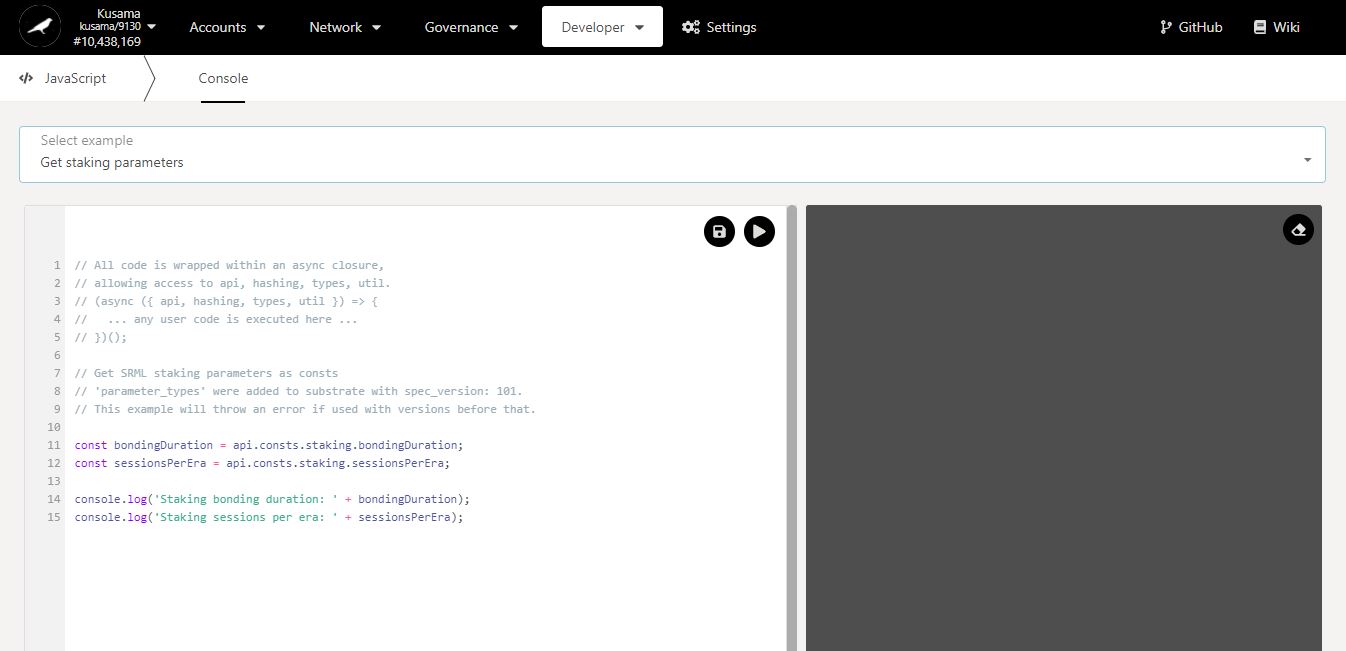
1. **Javascript: Interact with on-chain data through the console.**



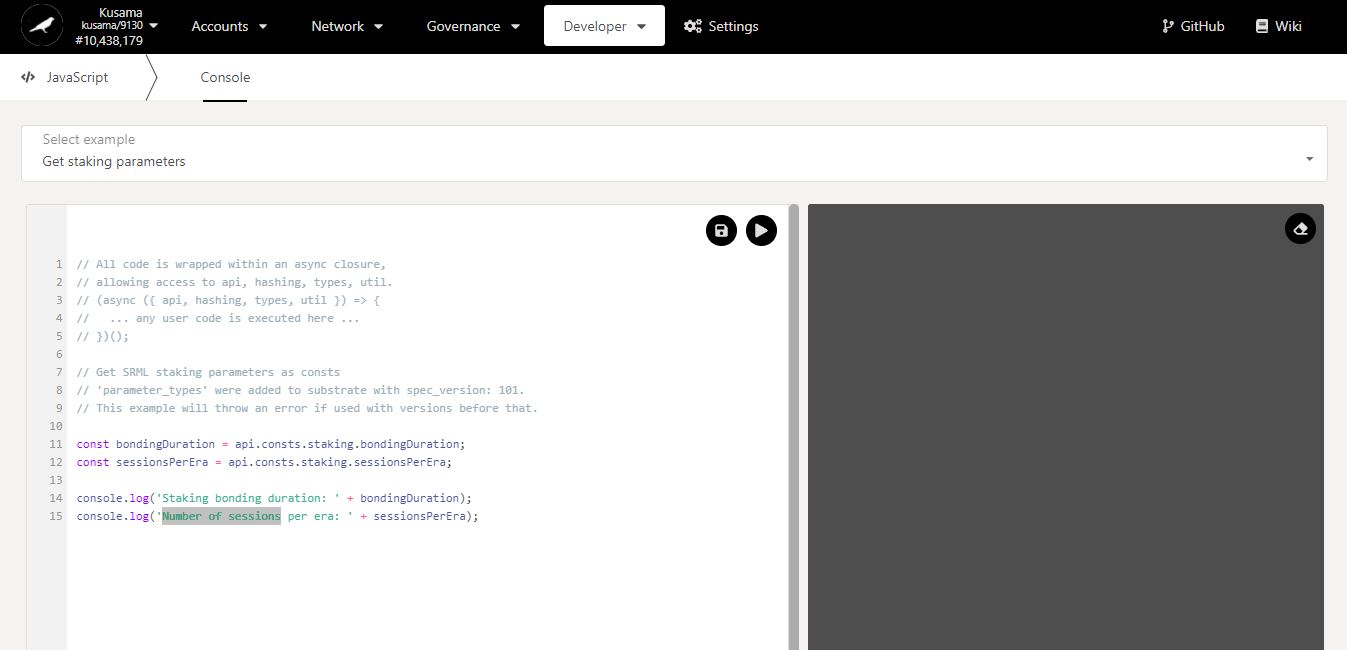
* 1. **View and submit code snippets.**



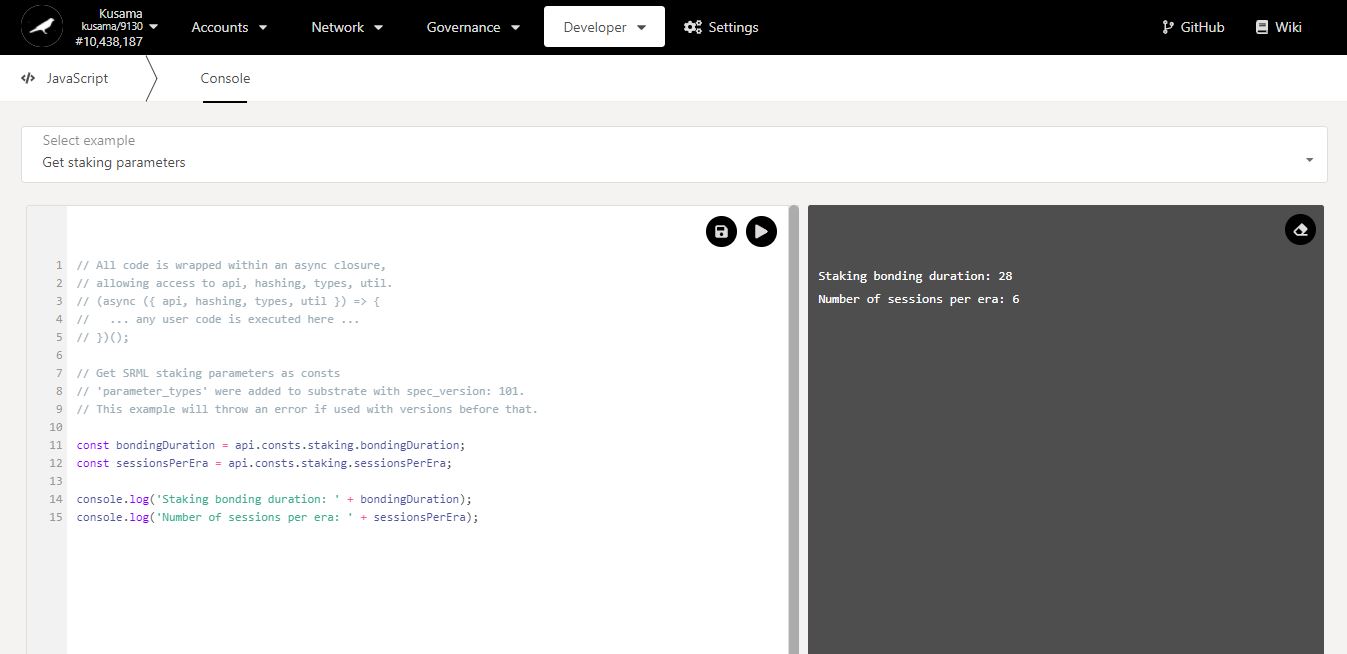
Interface for **executing JS scripts** on-chain in isolation.



1. Click on the **dropdown arrow** and select a **ready-made code snippet** to execute.



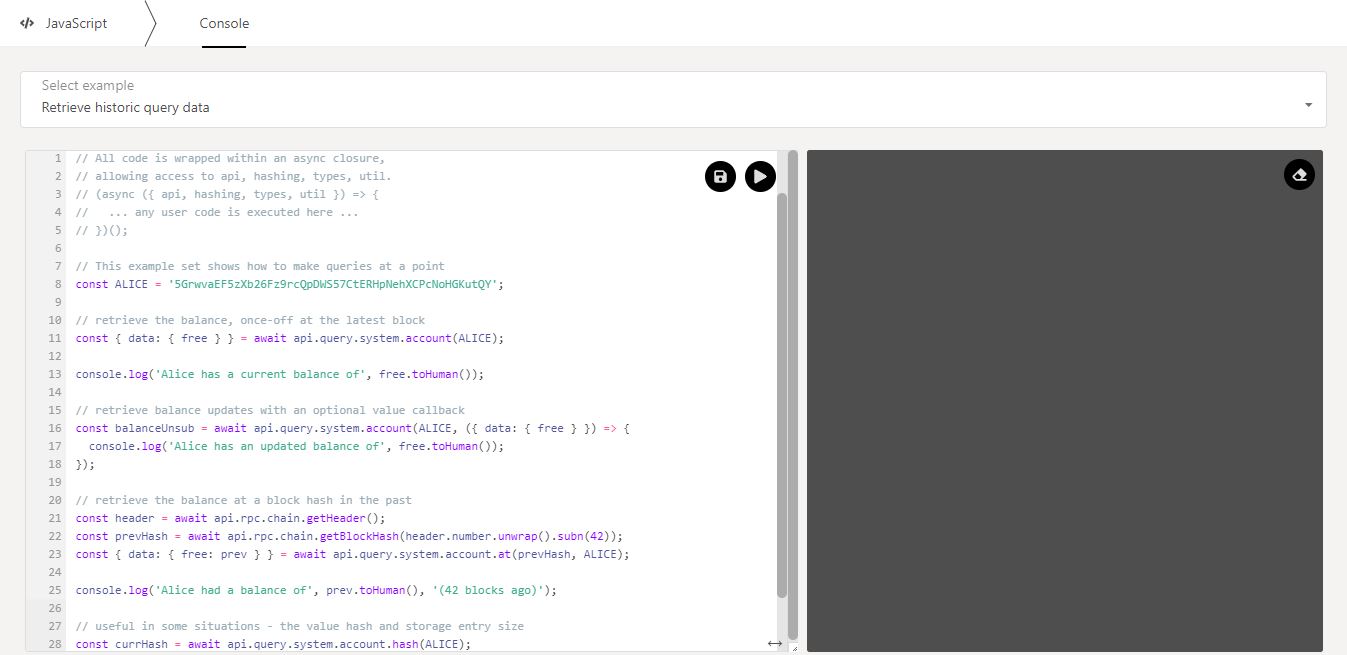
2. Change the **labels** (in green) for better legibility of the data.



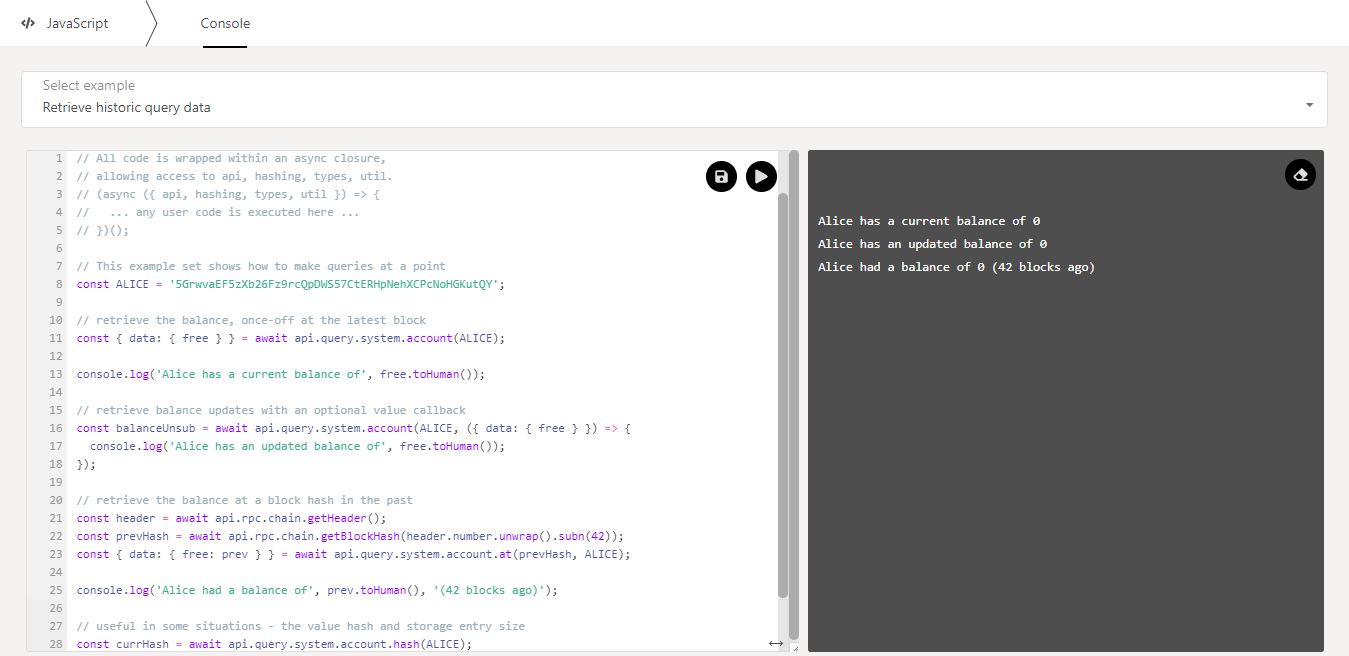
**4. Results are now available in the JS console!**

3. Click on the **play button** to run the script.

* 1. **Customise code snippets.**

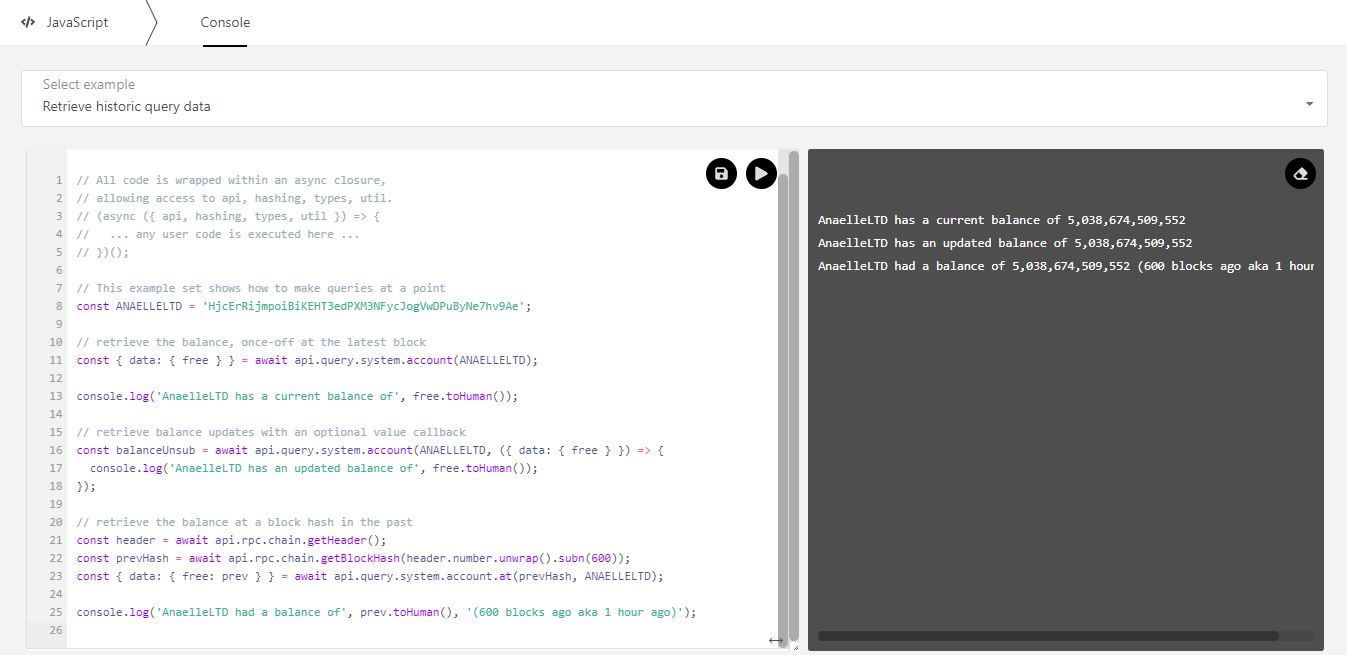


1. Click on the **dropdown arrow** and select a **ready-made code snippet** to customise.

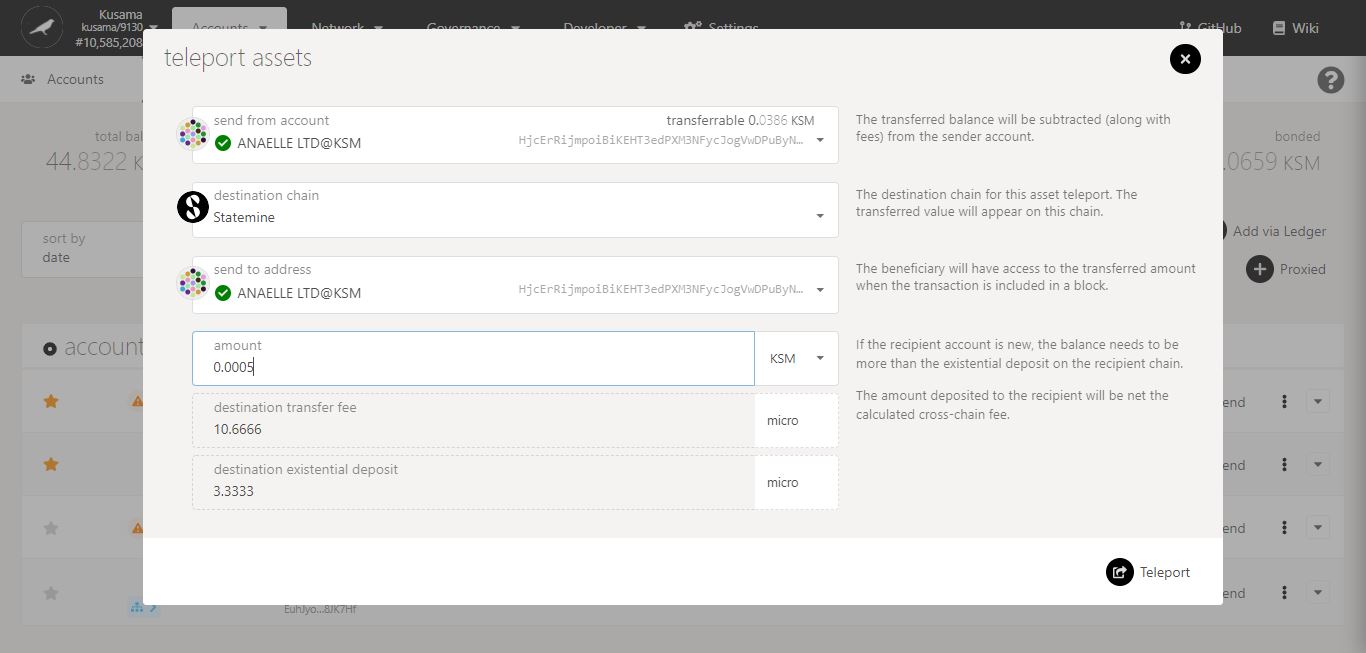


3. Click on the **eraser** button to clear the console.

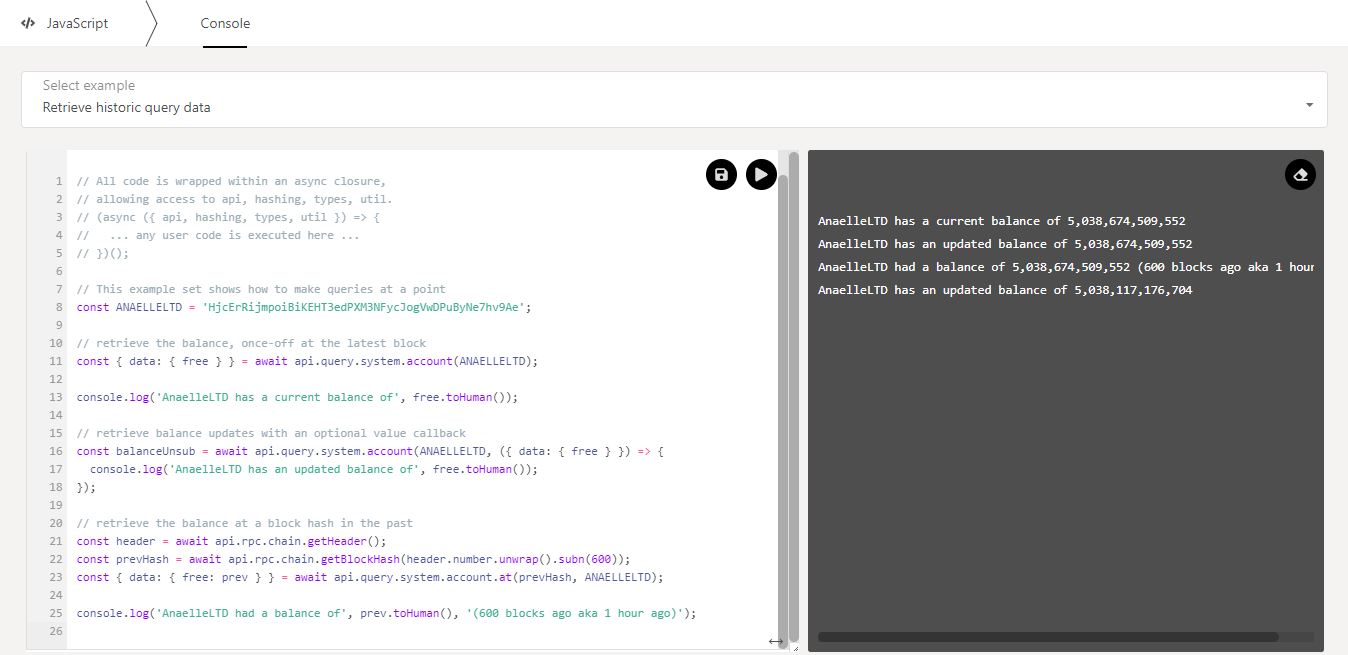
2. Click on the **play button** to run the script and show the results in the JS console.



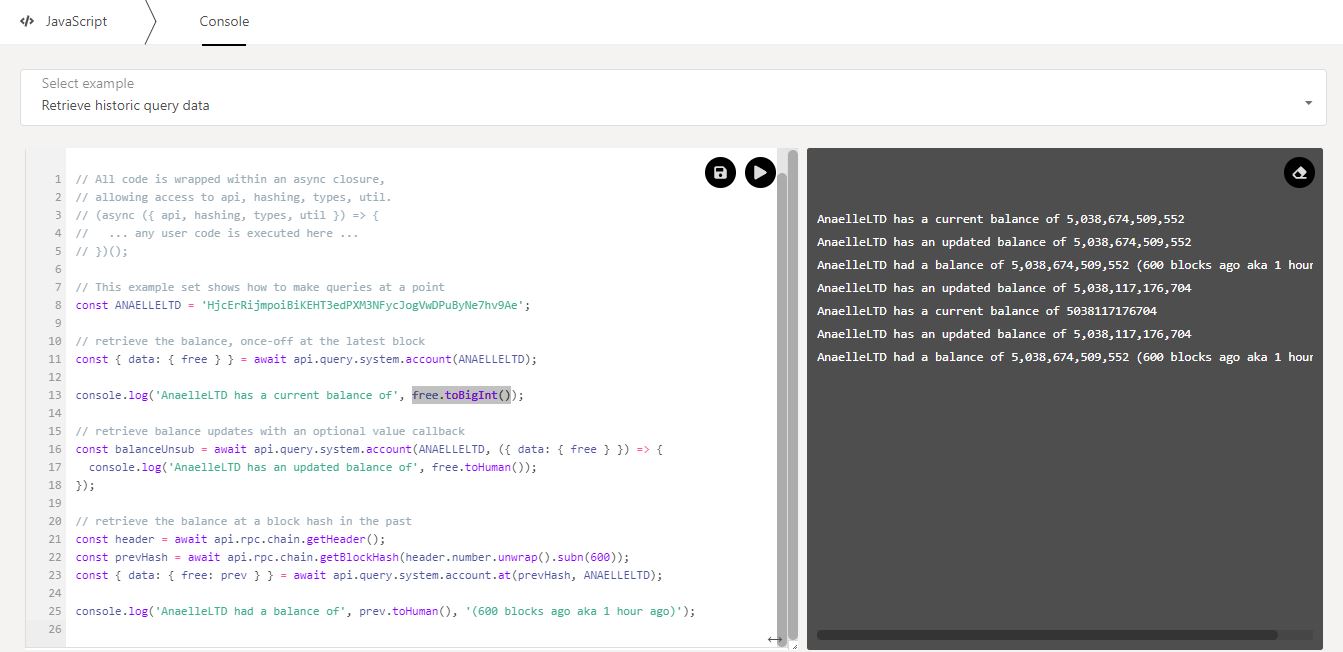
4. Change the **variables** (in blue) and the **labels** (in green) to customise the script, then run the script to show the results.



5. Submit with on-chain transactions that will further **test your custom script**.



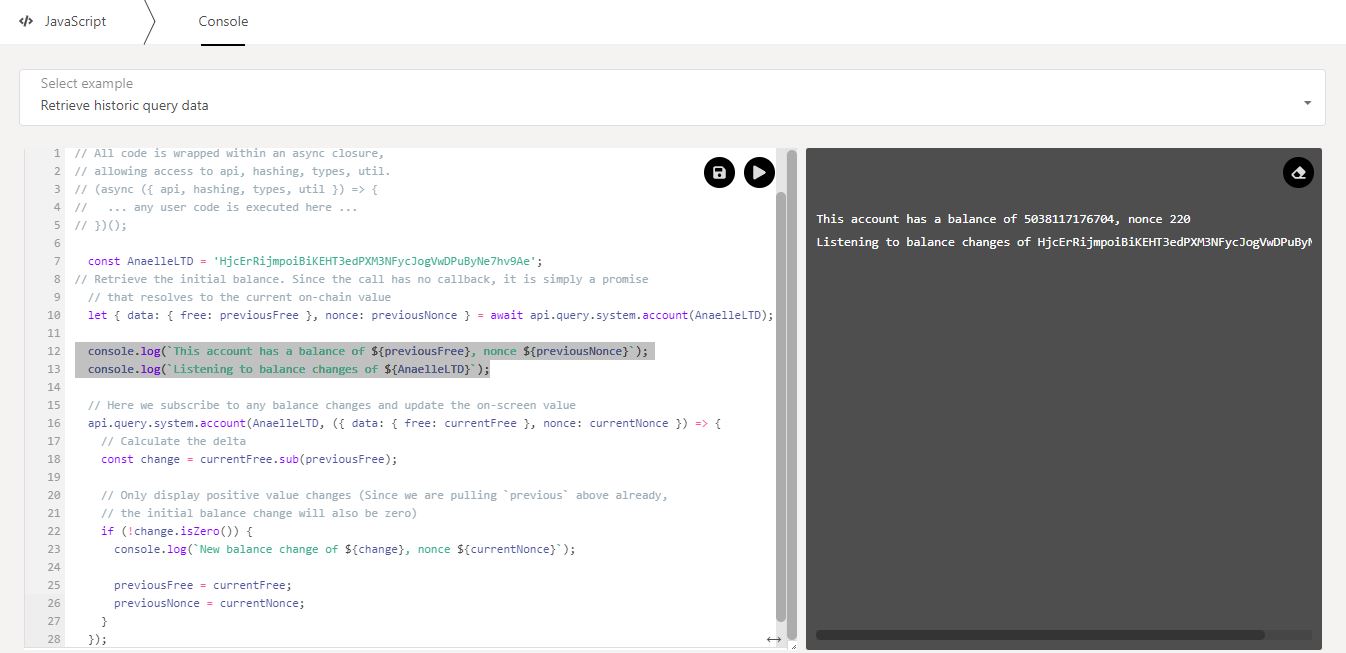
6. Check the **updated results**.



7. Change the **functions** (in purple) to further customise your script, then run the script to show the results.

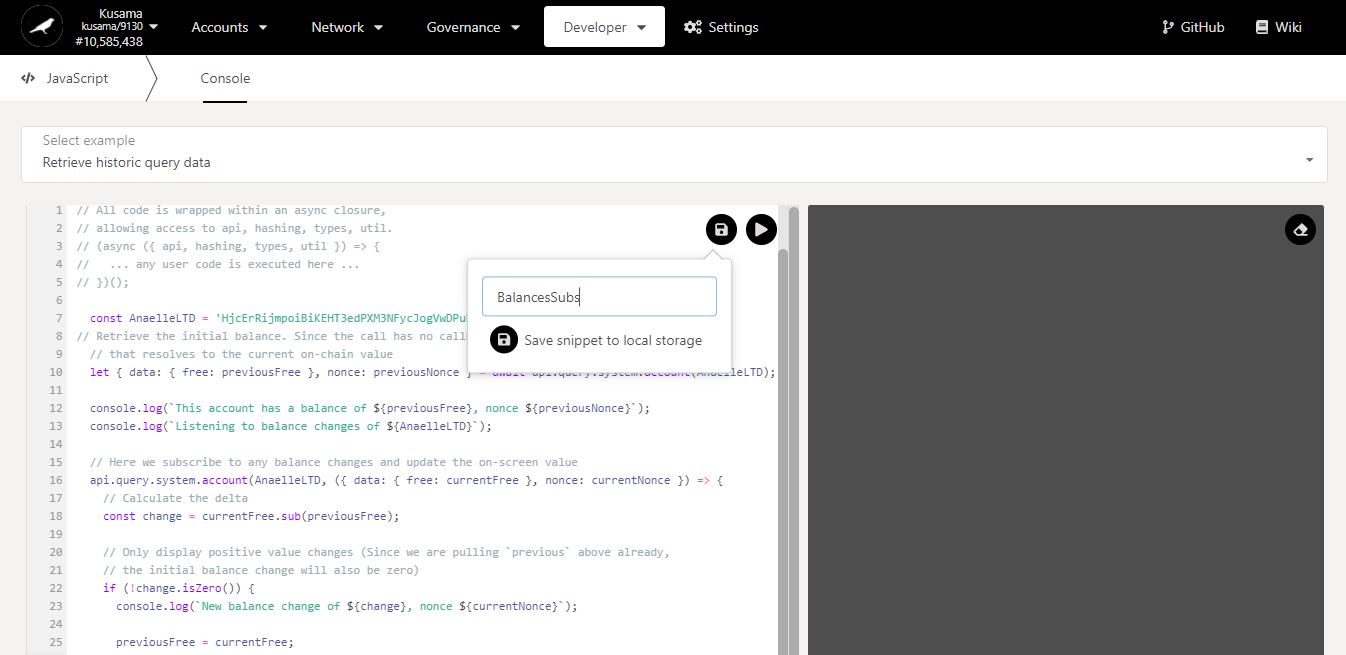
script and

* 1. **Back up code snippets.**



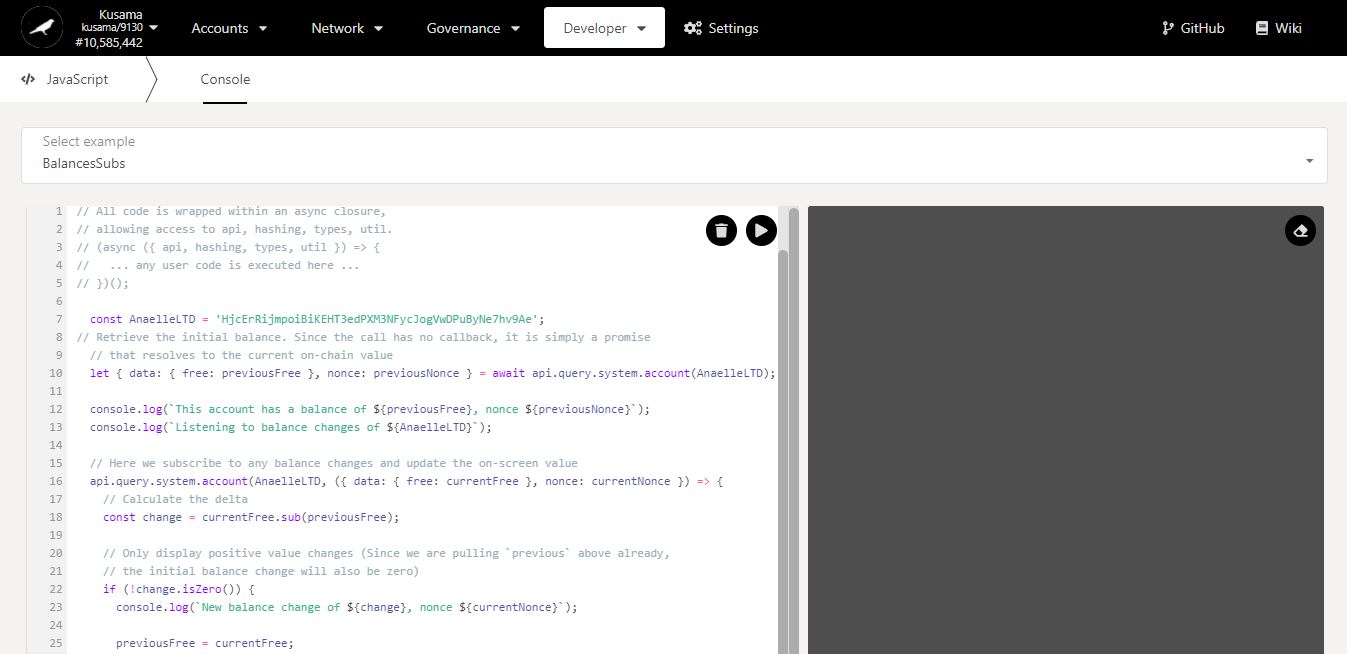
1. Add more **custom-made code** to your script, then run the script to show the results, and clear the console.

script and



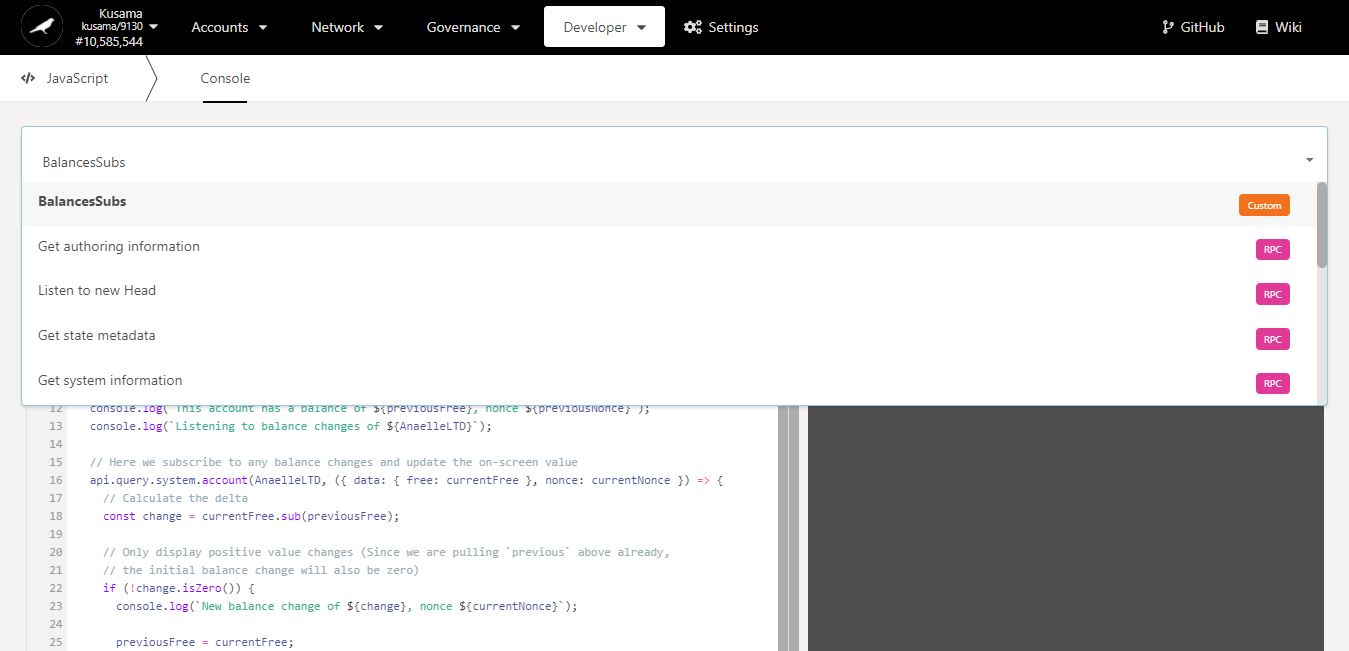
2. Click on the **save** button to back up your custom snippet locally.

3. Enter a name and click on **Save snippet** to finish the procedure.



5. Click on the **dropdown arrow** to check the list of available snippets.

**4. Your snippet has been renamed!**



**6. Your snippet is now (temporarily) stored in the dropdown menu!**