**INTRsupreme instructions:**

**CreateAccount()**

This creates a new user account whose address is derived from user’s ethereum wallet address. If no ethereum address, then derivation will come from user’s Solana wallet address. Balance is initialized to zero.

**CreateStake()**

This creates a new user stake account whose address is derived from the user’s account address, and the index whih represents the stake number. For example:

(user account address + 1) = stake account address #1

(user account address + 2) = stake account address #2

… etc

The URL hash will be produced by the user client with no need to make a query to database. Whatever the stake amount it, this is the amount. If stake accounts yield, then this amount cannot change, for we would need to track multiple times. So, to change a stake amount, a new stake account must be initialized. The time variable is determined by block time at instruction execution.

If no entity account exists, stake fails. (Any SS initial stake on non grey site will need to call CreateEntity() instead.) Entity account info is pulled and stakeneg, stakepos, and stakers are updated. If these variables (and time) pass threshold the proper flag is set in entity account.

User account stakes integer incremented.

**CreateEntity()**

This creates a new entity account whose address is derived from the entity’s url hash. It also creates the stake account for initial staker.

Entity account will have zero address (unclaimed) if SS is initial staker. Entity account will have hunter’s account address if claimed. Clientside will crawl preexisting stakes and pass stake data to entity account upon execution. ‘can-stake’ flag sets high.

Stake account address derived for caller, and adjusts account info accordingly.

**ACCOUNT**

**owner:** Pubkey, 32B

**vault:** Pubkey, 32B

**balance:** u128, 16B

**stakes:** u16, 2B

**success:** u16, 2B

**fail:** u16, 2B

**flags:** u16, 2B

**Size:** 88B

**Rent:** $0.1503

**STAKE**

**hash:** Pubkey, 32B

**amount:** u128, 16B

**time:** u64, 8B

**flags:** u16, 2B

**Size:** 58B

**Rent:** $0.1295

**Account definitions and types:**

**NOTES:**

**owner** is user’s Solana account public key

**vault** is Pubkey format of user’s ethereum wallet

**stakes** is the number of stake accounts a user controls

**success** is the number of successful stakes user has accumulated

**fail** is the number of failed stakes user has accumulated

**flags** this is a set of 16 bits for signaling and control

**hash** this is a 256b hash of grey area entity URL

… a map of these hashes to URLs is kept on server

**time** this is the chain time that a stake account was created

… yield / reward will be determined by this time

**hunter** this is the Pubkey of the bounty hunter’s account address

**netstake** this is the absolute total staked on entity

! rent is returned to caller when account is closed

! u128 presumes 18 decimals

! there is one account per entity claimed by bounty hunter

! unclaimed entities get zero address

**FLAGS:**

. account type 1 ENTITY

. account type 0

. stake mintotal threshold, high(over) low(under)

. stake mintime threshold, high(over) low(under)

. staker number threshold, high(over) low(under)

. settled? high(settled) low(not settled)

. settling? High(settling) low not settling)

. entity valence high(good) low(bad)

.

**ENTITY**

**hunter:** Pubkey, 32B

**stakeneg:** u128, 16B

**stakepos:** u128, 16B

**stakers:** u16, 2B

**time:** u64, 8B

**flags:** u16, 2B

**Size:** 76B

**Rent:** $0.1420

**FLAGS:**

. account type 0 STAKE

. account type 1

. stake valence high(url is good), low(url is bad)

.

.

**FLAGS:**

. account type 0 ACCOUNT

. account type 0

. bounty hunter? high(yes) low(no)

. connected to Ethereum? high(yes) low(no)

.

.

**SettleEntity()**

This is a simple instruction that sets the settled flag to high, and the valence flag to whatever the determination is.

The idea here is that this instruction will initiate an async style instruction that kicks off a voting round among BHs. During this time the settling flag is high but settled flag is low, and no new stakes can be made, nor can stakes be closed. After voting period has ended, these flags are updated, settling flag set low, so stakes can be closed out.

**CloseStake()**

This transfers remaining stake amount back to user account. If entity outcome flag was high, then reward fraction is applied to amount transferred. If low, slash is applied. Total time elapsed is also computed, and yield calculated and applied. Meanwhile, the final stake account in the user’s index of stake accounts replaces this account’s information (this resets the ordering for PDAs). Finally, the lamports in the final account are transferred back to GLOBAL which officially closes the stake account.

If outcome matches entity valence flag, then user account’s success counter is incremented. Likewise, the opposite for fail counter.

On the client side, every time an update function is called (ie each time stakes are scanned for changes, each stake’s associated entity account is checked for the settled flag high. If high, CloseStake is called.

**IdHunter()**

This function declares a user account a bounty hunter or not. Only the Interlock Network server will have ability to call this function, after 1) BH signs up/registers through splash page, or 2) BH loses reputation or reason to remain a BH.

**ENTITY**

**balance = 0**

**stakers++**

**stakepos += s**

Transfer = s

**ACCOUNT**

**balance = 0**

**stakes ++**

**CreateStake()**

**Staking Flow:**

**CreateAccount()**

**ACCOUNT**

**balance = s**

**STAKE**

**amount = s**