AIP-010: Simple Payment Service

lia@, seo@ 2021-02-02

Problem Definition

- Current deposit native feature doesn't support depositing on behalf of another user, and only the depositor can withdraw the fund.
- Depositors are "untrusted" by nature, meaning that the depositor cannot withdraw until a fixed lock-up time has passed.
- We need a new payment system where an admin accepts a payment off-chain and records the payment by transferring her balance to a service account.

Requirements

- A service admin can record the payment history of a customer
- A service admin can claim the payment
- The payment history should reflect the service payment's balance and thus service accounts' balance
- Total token circulation within the system should equal the sum of accounts' balances and the sum of service accounts' balances

Proposed Solution

- Introduce new predefined db paths:
    - /payments/{serviceName}
    - /service_accounts/payments/{serviceName}/{key}
- At /payments/{serviceName}, only service admin can write
- Upon a new payment entry, _pay or _claim native function is executed
    - At /payments/{serviceName}/{userAddress}/{paymentKey}/pays/{recordId}
        - /service_accounts/payments/{serviceName}/{userAddress}|{paymentKey}/balance += amount
        - /accounts/{serviceAdmin}/balance -= amount
    - At /payments/{serviceName}/{userAddress}/{paymentKey}/claims/{recordId}
        - /service_accounts/payments/{serviceName}/{userAddress}|{paymentKey}/balance -= amount
        - /accounts/{serviceAdmin}/balance += amount
- Total supply = (All balances under /accounts/{address}/balance) + (All balances under /service_accounts/{serviceName}/{serviceId}/{key}/balance)

- For claim requests, if `escrow_key` is specified, the payments are held in escrow instead of directly being transferred to the target

Payments:

```
/payments/collaborative_ai: {
  "0xUSER_ADDR": {
    "0": {
      "pays": {
        "12345678": {
          "amount": 1000,
          "pay_method": "card"
        },
      },
      "claims": {
        "23456789": {
          "amount": 500,
          "target": "0xTARGET_ADDR"
        },
        "34567890": {
          "amount": 500,
          "target": "0xTARGET_ADDR",
          "escrow_key": "abc"
        }
      },
    },
    "1": {
      "pays": {
        "12345678": {
          "amount": 100,
          "pay_method": "paypal"
        },
      }
    }
  },
  "config": {
    "admin": "0xADMIN_ADDR"
  }
}
```

Service accounts:

```
/service_accounts: {
  "payments": {
    "collaborative_ai": {
      "0xUSER_ADDR|0": {
        "balance": 10000000,
```

```
        "admin": "0xADMIN_ADDR_1"
      }
    },
    "afan": {
      "0xUSER_ADDR|0": {
        "balance": 100,
        "admin": "0xADMIN_ADDR_2"
      },
      "0xUSER_ADDR|1": {
        "balance": 10000,
        "admin": "0xADMIN_ADDR_3"
      }
    }
  },
  "deposit": {
    "consensus": {
      "0xUSER_ADDR": {
        "balance": 10000000,
        "admin": "0xADMIN_ADDR_4"
      }
    }
  }
}
```

Conclusion

- Provided schema for payment and service accounts.
- As future work, an escrow service can be added as a new type of payment.

Document History

Date	Who	Change	Notes
2021-02-02	lia@	Initial draft	
2021-03-02	lia@, seo@	Internal review	
2021-05-07	seo@	Published	