

Project presentation

# Python SDK for NexRes

---



By: Arindaam Roy, UC Davis  
Guided By: Prof. Mohammad Sadoghi, UC Davis



# Table of contents

**01**

## Motivation and Objectives

SDK for NFT marketplace

**02**

## Project Contributions

**03**

## Core Concepts

Transaction structure

**04**

## Demo

Creation and Transfer of Assets

**05**

## Architecture & Transaction flow

**06**

## Challenges and Future Work



**01**

---

# Motivation and Objectives

# SDK: a precursor to NFT Marketplace

- What is NexRes?
  - Next generation of ResilientDB (ResDB)
    - High Throughput Yielding Permissioned Blockchain Fabric
  - A consensus engine
    - core consensus protocol is based on a highly optimized PBFT
  - A key-value store with durable storage
  - Written in C++

# SDK: a precursor to NFT Marketplace

- What is needed for NFT Marketplace?
  - Easy way to create and transfer asset
  - Validation of transactions
  - Support for modern backend languages like python

**02**

---

# **Project Contributions**

# UTXO model on NexRes

## Client

### 1. Python SDK

- Prepares the Tx
- Signs/fulfills the Tx
- Sends the Tx over REST endpoints

### 2. KV interface

- REST endpoints in C++
- To post a Tx
- To get a Tx

## Server

### 3. Tx validation

- Signature
- Double spend
- Duplicate Tx
- etc

**03**

---

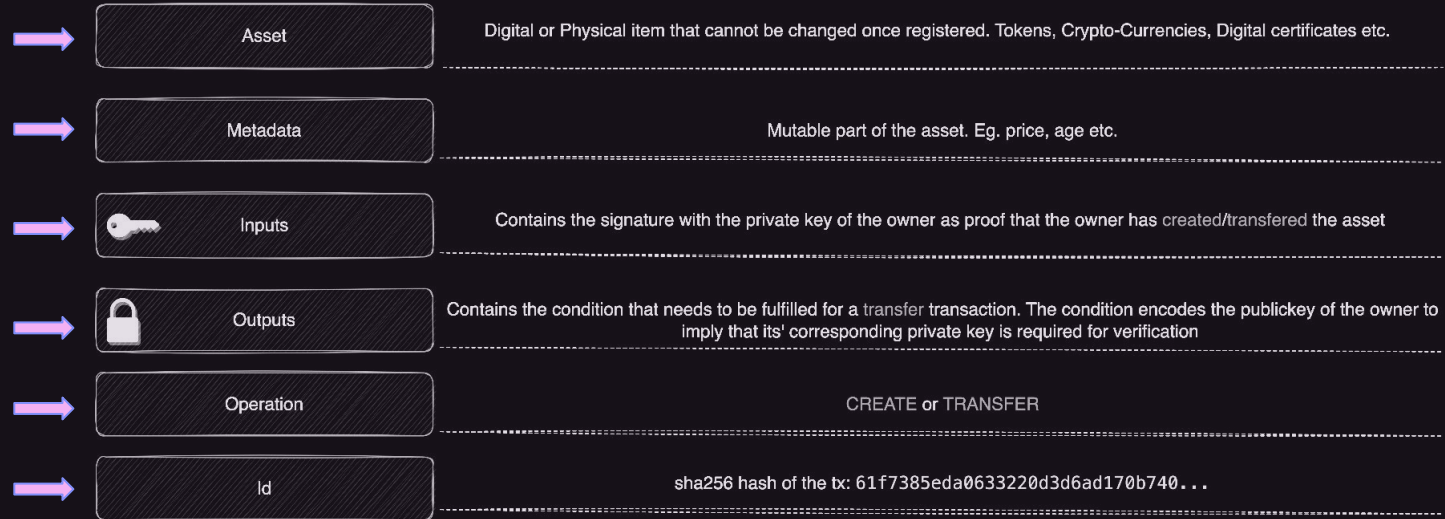
**Core concepts**

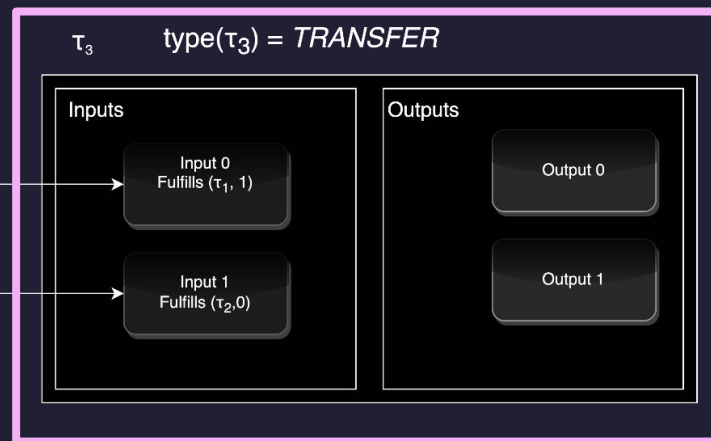
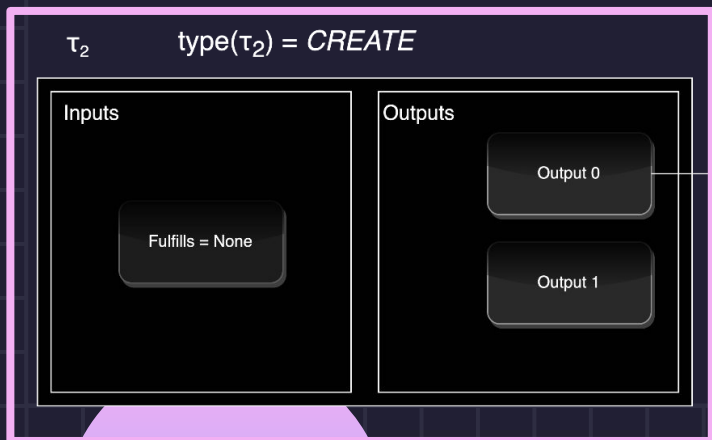
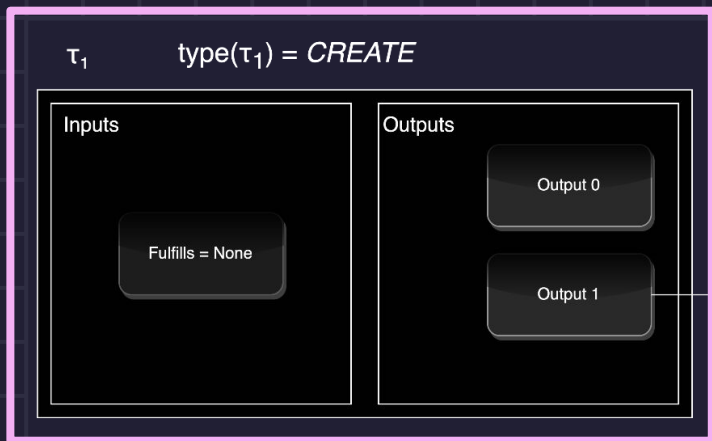


# Transactions (Tx)

- Enforces a UTXO model
- Encodes information such as:
  - Public keys (Owners)
  - Fulfillment of previous Tx
  - Asset info
- Inspired from BigChainDB transaction spec (BEP-13)

# Tx Structure





# Tx Validation

- NexRes validates the inputs of a Tx
- Ed25519 public-key signature to validate if the output of a tx is fulfilled by the correct owner
- Prevent double spend by checking if  $(id(Tx_i), index)$  is part of the input of any committed or enqueued Tx

**04**

---

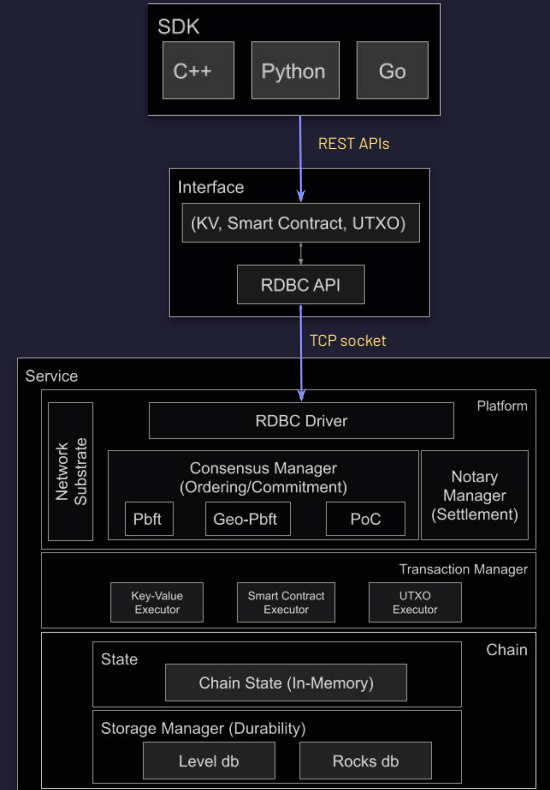
**Demo**

**05**

---

# Architecture & Tx Flow

# NexRes Architecture



# Tx Flow

Tx Preparation → Tx Fulfillment → Tx Verification → Tx Commitment

```
graph LR; A[Tx Preparation] --> B[Tx Fulfillment]; B --> C[Tx Verification]; C --> D[Tx Commitment]
```





**06**

---

# **Challenges and Future work**

# Challenges

- Securely storing private keys
  - The SDK can generate private and public keys but they need to be secure stored
- Validation is a python binding (makes it slow)
  - C++ does not have well maintained cryptoconditions libs

# Future work

- Validation in C++ (ongoing work)
- Using a persistence storage which allows for complex queries
- Requiring the signatures of both current and future owners for creation and transfer of assets
- Explore ResDB network with the sdk
- Package the SDK to PyPI



**Thank You!**