

# Notification Intro

- 说明

```
// 通知事件名定义
const enum PushEventType {
    AccountNew = 'account.new',           // App下有新的账号创建事件
    BalanceUpdate = 'balance.update',     // App下指定账号的余额发生改变
    事件
    TransactionCreated = 'transaction.created', // App下指定账号发生了提币(转账)交易事件
    TransactionConfirmed = 'transaction.confirmed' // App下与特定账号关联的交易被打包到链上事件
}

// 通知所属平台与关联的代币信息
const enum PushPlatform {
    BTC = 'bitcoin.btc',                 // Bitcoin平台下的btc
    ETH = 'ethereum.eth',                 // Ethereum平台下的eth
    OMNI_USDT = 'bitcoin.omni_usdt',      // Omni平台下的USDT
    ERC20_USDT = 'ethereum.erc20_usdt'    // ERC20平台下的USDT
}

// WebHook回调接收到的数据格式
type NotificationType = {
    type: PushEventType;                 // 事件类型
    platform: PushPlatform,              // 事件所属平台与代币信息
    data: any;                           // 特定事件会有与之想着的数据结构说明，如下(DataType)
}
```

## account.new #新建账号

bitcoin.btc

bitcoin.omni\_usdt

ethereum.eth

ethereum.erc20\_usdt

```
type DataType = {
    accountId: string;    // 账号Id
    address: string;      // 地址
}
```

## blanace.update #余额更新

---

bitcoin.btc

bitcoin.omni\_usdt

ethereum.eth

ethereum.erc20\_usdt

```
type DataType = {  
  accountId: string;    // 账号Id  
  address: string;      // 地址  
  // @note: 各平台的最小单位, 如Satoshi  
  balance: string;      // 余额信息  
}
```

## transaction.created # 创建交易

---

bitcoin.btc

bitcoin.omni\_usdt

ethereum.eth

ethereum.erc20\_usdt

```
type DataType = {  
  status: boolean;      // 交易是否成功,  
  accountId: string;    // 账号Id  
  address: string;      // 地址  
  serial: number;       // 请求序号, Http返回也有此请求序号  
  // @note: status=true时, 此字段有效  
  txId: string;         // 交易Id(此时交易还未确认, 刚广播到网络中)  
  // @note: status=false时, 此字段有效  
  error: string;        // 交易失败时的错误信息  
}
```

## HTTP Desposit Response

```

type Response = {
  success: boolean;           // true
  data: {                     // HTTP Response数据
    status: boolean;          // 请求处理状态
    serial: number;           // 请求序号, 与Notification中的序号对应
    // @note: status=true时, 此字段有效
    txId: string;             // 交易Id
    // @note: status=false时, 此字段有效
    error: string;            // 请求处理失败的错误信息
  }
}

```

## transaction.confirmed && bitcoin.btc(比特币交易确认)

```

// 交易方向
type const enum TransactionDirection = {
  In = 'In',                 // 输入方向(other -> self)
  Out = 'Out',               // 输出方向(self -> other)
}

```

### bitcoin.btc

```

type VIn = {
  address: string;           // 交易输入地址
  amount: string;            // 交易输入金额
}
type VOut = {
  address: string;           // 交易输出地址
  amount: string;            // 交易输出金额
}

type Transaction = {
  txId: string;              // 交易Id
  blockHeight: number;       // 交易打包高度
  blockTime: number;         // 区块打包时间
  fee: string;               // 交易费
  vIns: VIn[],               // 交易输入
  vOuts: VOut[]              // 交易输出
}

type DataType = {
  accountId: string;         // 账号Id
  address: string;           // 比特币地址
}

```

```

    direction: TransactionDirection; // 交易方向
    transaction: Transaction;        // 交易详情
}

```

## bitcoin.omni\_usdt

```

type Transaction = {
    txId: string;                // 交易Id
    blockHeight: number;         // 交易打包高度
    blockTime: number;           // 交易打包区块时间
    propertyId: number;          // Omni属性值, 一个Omni Token对应一个id
    version: number;             // 交易版本
    typeInt: number;             // 交易类型(int)
    fee: string;                 // 交易费
    sending: string;             // 交易发送者
    reference: string;           // 交易接收者
    amount: string;              // 交易金额
}

type DataType = {
    accountId: string;           // 账号Id
    address: string;             // 比特币地址
    direction: TransactionDirection; // 交易方向
    transaction: Transaction;    // 交易详情
}

```

## ethereum.eth

```

type Transaction = {
    txId: string;                // 交易Id
    blockHeight: string;         // 交易打包高度
    nonce: string;               // 发送交易次数
    sender: string;              // 交易发送者
    recipient: string;           // 交易接收者
    amount: string;              // 交易金额
}

type DataType = {
    accountId: string;           // 账号Id
    address: string;             // 以太坊地址
    direction: TransactionDirection; // 交易方向
    transaction: Transaction;    // 交易详情
}

```

## ethereum.erc20\_usdt

```
type Transaction = {  
  txId: string;           // 交易Id  
  blockHeight: string;    // 交易打包高度  
  fee: string;            // 交易费  
  sender: string;         // 交易发送者  
  recipient: string;      // 交易接收者  
  amount: string;         // 交易金额  
}  
  
type DataType = {  
  accountId: string;      // 账号Id  
  address: string;        // 以太坊地址  
  direction: TransactionDirection; // 交易方向  
  transaction: Transaction; // 交易详情  
}
```