TxPipe Docs

# Parties

Parties are one of the foundational concepts in Tx3. They represent the various actors that interact with your blockchain protocol, such as users, contracts, or services.

## Why Parties Matter

In UTxO blockchains like Cardano, addresses control UTxOs. Rather than hardcoding addresses, Tx3 uses the concept of parties to:

1. **Abstract implementation details** - Separating roles from specific addresses
2. **Improve readability** - Making transactions easier to understand
3. **Enable configuration** - Allowing the same protocol to work with different addresses
4. **Support parameterization** - Letting transactions work with dynamic parties

## Party Definition Syntax

The basic syntax for defining a party is:

```
party {Name};
```

Where `Name` is an alphanumeric identifier (starting with a letter) that represents this party throughout your protocol.

## Simple Party Examples

```
// Basic parties representing different roles
party User;           // End user of the protocol
party Treasury;       // Treasury that holds protocol funds
party Oracle;         // Data provider
```

# Usage

Parties define the “who” in transactions - who provides inputs and who receives outputs.

# Input Sources

Parties specify who controls the UTxOs that will be consumed:

```
party User;
// User is providing Ada as input
tx deposit(amount: Int) {
    // UTxO controlled by the User party
    input user_tokens {
        from: User,
        min_amount: Ada(amount) + fees,
    }

    // Rest of transaction...
}
```

# Output Recipients

Parties specify who will receive newly created UTxOs:

```
party Treasury;
// Treasury is receiving protocol fees
tx fund() {
  // Rest of transaction...
  output {
    to: Treasury,        // UTxO will be sent to Treasury's address (which
can be a script address as well!)
    amount: Ada(fee_amount),
  }
}
```

# Complex Party Relationships

Real-world protocols typically involve multiple parties with different roles:

```
party EscrowContract;
party Buyer;
party EscrowProvider;
party Seller;

tx escrow_payment(
    seller_item_id: Bytes,
    price: Int,
    escrow_fee: Int,
) {
    input payment {
        from: Buyer,
        min_amount: Ada(price + escrow_fee) + fees,
    }

    output locked_payment {
        to: EscrowContract,
        amount: Ada(price),
        datum: EscrowState {
            buyer: Buyer,
            seller: Seller,
            item_id: seller_item_id,
            amount: price,
        }
    }

    output {
        to: EscrowProvider,
        amount: Ada(escrow_fee),
    }

    output {
        to: Buyer,
        amount: payment - Ada(price + escrow_fee) - fees,
    }
}
```

```
}
```

# Party Properties and Methods

Parties have properties and methods that can be accessed in transactions:

```
// Access party's address directly
User;

// In Cardano, get party's stake credential
Treasury.stake_credential;

// In Cardano, get party's payment credential
Treasury.payment_credential;
```

# Best Practices for Using Parties

1. **Use descriptive names** - Names like `LiquidityProvider` are clearer than `Party1`
2. **Consistent roles** - Each party should have a well-defined role in the protocol
3. **Security considerations** - Be clear about which parties need to authorize actions
4. **Documentation** - Document the responsibilities and expectations for each party