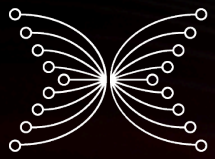
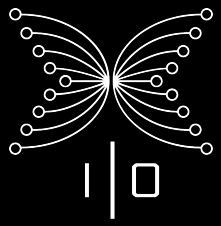


INPUT | OUTPUT

Contexto de la Transacción



01

Script Context

Contexto de la Tx : Qué contiene el contexto del script?

```
ScriptContext {  transaction: Transaction ,  redeemer: Redeemer ,  info: ScriptInfo }
```

Contexto de la Tx : Qué contiene el contexto del script?

```
ScriptContext { transaction: Transaction , redeemer: Redeemer , info: ScriptInfo }
```

- `transaction: Transaction` : Información sobre la transacción que ejecuta script/validador. Igual para todos los validadores de la transacción (en caso de haber más de uno).

Contexto de la Tx : Qué contiene el contexto del script?

```
ScriptContext {  transaction: Transaction ,  redeemer: Redeemer ,  info: ScriptInfo }
```

- `redeemer: Redeemer` : Conjunto de **datos arbitrarios** proporcionados por el usuario al interactuar con un contrato inteligente. Usualmente, el redeemer contiene **instrucciones o argumentos** que determinan cómo debe comportarse el contrato en esa transacción, dependiendo de la lógica programada en el script.

Contexto de la Tx : Qué contiene el contexto del script?

```
ScriptContext {  transaction: Transaction ,  redeemer: Redeemer ,  info: ScriptInfo }
```

- `info: ScriptInfo` : Ya hablamos de esto en clases anteriores. Único para cada validador de la transacción.

Contexto de la Tx : Qué contiene el contexto del script?

```
ScriptContext {  transaction: Transaction ,  redeemer: Redeemer ,  info: ScriptInfo }
```

- `transaction: Transaction` : Información sobre la transacción que ejecuta script/validador. Igual para todos los validadores de la transacción (en caso de haber más de uno).

- `redeemer: Redeemer` : Conjunto de **datos arbitrarios** proporcionados por el usuario al interactuar con un contrato inteligente. Usualmente, el redeemer contiene **instrucciones o argumentos** que determinan cómo debe comportarse el contrato en esa transacción, dependiendo de la lógica programada en el script.

- `info: ScriptInfo` : Ya hablamos de esto en clases anteriores. Único para cada validador de la transacción.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Entradas (Consumidas):

Lista de todos los UTxO que consume la transacción.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Entradas de referencia:

Lista de todos los UTXO que la Tx **lee sin consumir**. Esto permite a los scripts acceder al datum o al valor sin tener que gastar el UTXO.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Salidas (Creadas):

Lista de nuevos UTXOs que la transacción está creando.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Tarifa:

La cantidad total de **Lovelace** (la unidad mínima de ADA), que hay que pagar al nodo por validar y procesar la Tx.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Acuñado/Quemado:

La cantidad de **tokens nativos** que la Tx está **creando** (minteado) o **destruyendo** (quemando).

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Certificados:

Lista de certificados atestando una operación (registrar una pool, delegar, gobernanza…)

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Retiros:

Lovelace que están retirando en esta transacción,(de las cuentas de stake asociadas a una **Credential**). Se ordenan de forma ascendente por credencial.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Rango de validez:

Un intervalo de tiempo POSIX, medido en milisegundos desde el 01-01-1970 a las 00:00:00 (UTC). Durante el cual la Tx es válida.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Firmantes extra:

Lista de hashes públicos que firmaron la transacción.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Redeemers (Argumentos de Script):

Lista de pares que contienen los propósitos de todos los scripts de la Tx con sus respectivos redeemers. Se ordenan de forma ascendente por el propósito del script.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Datums (Datos On-chain):

Diccionario que relaciona los hashes de datums con los datums adjuntados a la transacción.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

ID de Transacción:

Identificador único de la transacción.
Hash de la transacción serializada.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Votos (Gobernanza):

Lista que representa los **votos emitidos** por diferentes partes (*Voter*) para acciones de gobernanza específicas (*GovernanceActionId*).

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GgovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Procedimientos de propuesta (Gobernanza):

Lista de **propuestas** de gobernanza incluidas en la transacción (ej. proponer un cambio en un parámetro del sistema).

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Tesorería actual:

El **saldo** de Lovelace en la Tesorería de Cardano **antes** de que se aplique esta transacción.

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: List<Input>,
  reference_inputs: List<Input>,
  outputs: List<Output>,
  fee: Lovelace,
  mint: Value,
  certificates: List<Certificate>,
  withdrawals: Pairs<Credential, Lovelace>,
  validity_range: ValidityRange,
  extra_signatories: List<VerificationKeyHash>,
  redeemers: Pairs<ScriptPurpose, Redeemer>,
  datums: Dict<DataHash, Data>,
  id: TransactionId,
  votes: Pairs<Voter, Pairs<GovernanceActionId, Vote>>,
  proposal_procedures: List<ProposalProcedure>,
  current_treasury_amount: Option<Lovelace>,
  treasury_donation: Option<Lovelace>,
}
```

Donación a la tesorería:

Cantidad de monedas a donar a la tesorería, (ej. para devolver dinero a la tesorería después de una acción de gobernanza).

Contexto de la Tx : Qué contiene `Transaction` ?

```
Transaction {
  inputs: [],
  reference_inputs: [],
  outputs: [],
  fee: 0,
  mint: assets.zero,
  certificates: [],
  withdrawals: [],
  validity_range: interval.everything,
  extra_signatories: [],
  redeemers: [],
  datums: dict.empty,
  id: #"0000000000000000000000000000000000000000000000000000000000000000",
  votes: [],
  proposal_procedures: [],
  current_treasury_amount: None,
  treasury_donation: None,
}
```

INPUT | OUTPUT

Preguntas?