Payment Networks

Rules for how a payment should be processed and detected

A payment network is a set of rules defining how a payment should be processed and detected for a Request. It specifies:

1. Information required at request creation to enable payment detection
2. The payment method
3. The process for determining the balance (amount paid)

Reference-based Payment Networks (Recommended)

Reference-based payment networks use a payment reference to link payments to the corresponding Request. They process payments via payment proxy smart contracts deployed across a wide variety of supported chains. These contracts serve two key functions:

1. They forward the payment to the payment recipient's address.
2. They emit an event containing the payment amount and the payment reference.

The payment reference is a unique identifier derived from the Request ID and payment recipient address. This derivation happens off-chain, not in the smart contract itself.

It's important to note that the payment recipient's address can be different from the payee's identity address that typically signs to create the request.

The payment proxy smart contracts do not perform error checking. It is the responsibility of the application using the Request Network to craft the payment transaction with the correct amount and payment reference.

A payment subgraph indexes events from the payment proxy smart contracts, enabling efficient payment detection and balance calculation for each request.

Please note that Reference-based payment networks inherit from the declarative payment network. This means that all reference-based requests can also be paid using declarative payments, providing flexibility in payment methods.

ERC20 Fee Proxy Contract

This payment network is used for direct ERC20 token payments.

Example of creating a request with an ERC20 Fee Proxy Contract payment network:

```
const erc20Request = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ERC20_FEE_PROXY_CONTRACT,
    parameters: {
      paymentNetworkName: 'sepolia',
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '0',
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ERC20,
      value: '0xdAC17F958D2ee523a2206206994597C13D831ec7', // USDT on
      network: 'mainnet'
    },
    expectedAmount: '1000000', // 1 USDT (6 decimals)
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

For details on how to use the ERC20 Fee Proxy, see the Quickstart - Browser or Quickstart - Node.js

ETH Fee Proxy Contract ("Native Payment")

This payment network is used for direct ETH (or native token) payments on Ethereum and EVM-compatible chains.

Example of creating a request with an ETH Fee Proxy Contract payment network:

```
const nativeRequest = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ETH_FEE_PROXY_CONTRACT,
    parameters: {
      paymentNetworkName: 'mainnet',
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '1000000000000000', // 0.001 ETH fee
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ETH,
      value: 'ETH',
      network: 'mainnet'
    },
    expectedAmount: '1000000000000000000', // 1 ETH (18 decimals)
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

This payment network allows for native token payments with an optional fee mechanism. It's suitable for ETH payments on Ethereum mainnet, or native token payments on other EVM-compatible chains like Polygon's MATIC or Binance Smart Chain's BNB.

For details on how to use the ETH Fee Proxy, see Native Payment

Any-to-ERC20 Proxy Contract "ERC20 Conversion Payments"

This payment network allows for "ERC20 Conversion Payments", where the payment currency is an ERC20 token that is different from the request currency. This is most commonly used to denominate the request in fiat like USD but settle the payment in ERC20 tokens like USDC or USDT. This is useful for accounting because most people use fiat currency as their unit of account.

Key features:

- Supports payments in any ERC20 token for requests created in various currencies
- Uses on-chain oracles for real-time currency conversion
- Includes a fee mechanism similar to the ERC20 Fee Proxy Contract

Example of creating a request with an Any-to-ERC20 Proxy Contract payment network:

```
const erc20ConversionRequest = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ANY_TO_ERC20_PROXY,
    parameters: {
      paymentNetworkName: 'mainnet',
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '100', // Fee in request currency
      acceptedTokens: ['0x6B175474E89094C44Da98b954EedeAC495271d0F'],
      maxRateTimespan: 1800, // 30 minutes
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ISO4217,
      value: 'USD'
    },
    expectedAmount: '1000000', // 1000.00 USD (2 decimals)
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

In this example, the request is created in USD, but can be paid using DAI (or any other specified ERC20 token). The conversion rate is determined at the time of payment using on-chain oracles.

Conversion is different from Swap-to-Pay. For details see Difference between Conversion, Swap-to-Pay, and Swap-to-Conversion

For details on how to use the Any-to-ERC20 Conversion Proxy Contract, see [Conversion Payment]

Any-to-ETH Proxy Contract "Native Conversion

payment currency is ETH (or native token) for requests denominated in other currencies, usually fiat. This is most commonly used to denominate the request in fiat like USD but settle the payment in native tokens like ETH on Ethereum or POL on Polygon PoS. This is useful for accounting because most people use fiat currency as their unit of account.

Key features:

- Supports payments in ETH (or native token) for requests created in various currencies
- Uses on-chain oracles for real-time currency conversion
- Includes a fee mechanism similar to the ETH Fee Proxy Contract

Example of creating a request with an Any-to-ETH Proxy Contract payment network:

```
const nativeConversionRequest = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ANY_TO_ETH_PROXY,
    parameters: {
      paymentNetworkName: 'mainnet',
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '100', // Fee in request currency
      maxRateTimespan: 1800, // 30 minutes
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ISO4217,
      value: 'USD'
    },
    expectedAmount: '1000000', // 1000.00 USD (2 decimals)
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

In this example, the request is created in USD but can be paid using ETH. The conversion rate is determined at the time of payment using on-chain oracles.

Conversion is different from Swap-to-Pay. For details see Difference between Conversion, Swap-to-Pay, and Swap-to-Conversion

For details on how to use the Any-to-ETH Proxy Contract, see Conversion Payment

ERC20 Transferable Receivable

ERC20 Transferable Receivable allows requests to be minted as NFTs that can be transferred or sold. When the payment is processed, the owner of the NFT receives the payment.

Example of creating an ERC20 Transferable Receivable request:

```
const transferableRequest = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ERC20_TRANSFERABLE_RECEIVA
    parameters: {
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '0',
      network: 'mainnet',
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ERC20,
      value: erc20TokenAddress,
      network: 'mainnet'
    },
    expectedAmount: '1000000000000000000', // 1 token with 18 decimal
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

For details on how to use ERC20 Transferable Receivables, see Transferable Receivable Payment

Declarative Payment Network

The declarative payment network allows for manual declaration of payments and refunds. It's particularly useful for currencies or payment methods not directly supported by Request Network like traditional finance payment rails, unsupported web3 payment protocols, and unsupported chains like Solana or Tron.

Example of creating a request with a declarative payment network:

```
const request = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.DECLARATIVE,
    parameters: {
      paymentInfo: {
        IBAN: 'FR7630006000011234567890189',
        BIC: 'BNPAFRPP'
      },
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ISO4217,
      value: 'EUR'
    },
    expectedAmount: '100000', // 1000.00 EUR (2 decimals)
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

For details on how to use Declarative Payments, See Declarative Payment

Meta Payment Network

The Meta Payment Network allows you to specify multiple potential payment networks for a single request. The payment can be settled by any one of the sub-payment networks.

For details on how to use Meta Payment Network, see Meta Payments

ERC777 Stream Payment Network "Streaming Payment"

ERC777 Streaming Payment routes payments through the ERC20 Fee Proxy payment network, allowing for continuous, streaming payments.

Example of creating an ERC777 Streaming Payment request:

```
const streamingRequest = await requestNetwork.createRequest({
  paymentNetwork: {
    id: Types.Extension.PAYMENT_NETWORK_ID.ERC777_STREAM,
    parameters: {
      paymentAddress: paymentRecipientAddress,
      feeAddress: feeRecipient,
      feeAmount: '0',
      network: 'mainnet',
      tokenAddress: erc777TokenAddress,
    },
  },
  requestInfo: {
    currency: {
      type: RequestLogicTypes.CURRENCY.ERC777,
      value: erc777TokenAddress,
      network: 'mainnet'
    },
    expectedAmount: '1000000000000000000', // 1 token with 18 decimal
    payee: payeeIdentity,
    payer: payerIdentity,
  },
  signer: payeeIdentity,
});
```

For details on how to use the ERC777 Stream Payment Network, see Streaming Payment

Other Payment Networks

Address-based Payment Networks (Not Recommended)

These networks require a unique payment recipient address for each request. The balance is computed from all inbound transfers to this address. This method is not recommended due to its limitations and potential for errors.

ETH Input Data Payment Network (Deprecated)

This network used the `call data` field of Ethereum transactions to tag and detect payments. It has been deprecated in favor of the other reference-based payment networks that use payment proxy smart contracts.

ERC20 Proxy and Ethereum Proxy (Superseded)

These payment networks have been superseded by the ERC20 Fee Proxy and ETH Fee Proxy networks respectively. The only difference is that the ERC20 Proxy and Ethereum Proxy don't include the service fee mechanism. For most use cases, it's recommended to use the Fee Proxy versions with the fee set to 0 if no fee is required.

Advanced Payment Types

Advanced payment types are built on top of payment networks and provide additional functionality or flexibility.

The Advanced Payment Types are *not* Payment Networks. You cannot create a request with one of these payment types in the `paymentNetwork` property.

Here are some of the advanced payment types available:

Batch Payments

Batch payments allow you to pay multiple requests in the same transaction. This is supported for the following payment networks:

- ERC20 Fee Proxy
- ETH Fee Proxy
- Any-to-ERC20 Proxy "ERC20 Conversion Payments"
- Any-to-ETH Proxy "ETH Conversion Payments"
- ERC20 Proxy networks

See [Batch Payment] for additional details.

ERC20 Swap-to-Pay Payments

Swap-to-Pay payments execute a swap immediately before routing the payment through the ERC20 Fee Proxy payment network.

Swap-to-Pay is different from Conversion. For details see Difference between Conversion, Swap-to-Pay, and Swap-to-Conversion

See Swap-to-Pay Payment for additional details

ERC20 Swap-to-Conversion

Swap-to-Conversion is the combination of Conversion and Swap-to-Pay.

Swap-to-Conversion executes a swap in Uniswap V2 immediately before routing the payment through the Any-to-ERC20 Payment Network.

See Swap-to-Conversion Payment for additional details.

Swap-to-Pay is different from Conversion. For details see Difference between Conversion, Swap-to-Pay, and Swap-to-Conversion

ERC20 Escrow Payment

ERC20 Escrow Payment allows for escrow functionality in payments.

The Request Network Escrow lacks arbitration and is susceptible to deadlock in the case of a dispute.

See Escrow Payment for additional details.

Difference between Conversion, Swap-to-Pay, and Swap-to-Conversion

Conversion is different from Swap-to-pay.

- In a conversion payment, the request is denominated in currency A, the payer sends currency B and the payee receives currency B.
- In a Swap-to-pay payment, the request is denominated in currency A, the payer sends currency B and the payee receives currency A.

They can be combined into Swap-to-Conversion.

- In a swap-to-conversion payment, the request is denominated in currency A, the payer sends currency B and the payee receives currency C.

For more detailed information on specific payment networks or to contribute new implementations, please refer to our GitHub repository or join our Discord community.

Previous
SDK and Request Node Overview

Next
Private Requests using Encryption

Last updated 2 months ago