0xSwap Partner API Documentation

Overview

The 0xSwap Partner API allows third-party services to integrate cryptocurrency exchange functionality directly into their platforms. This RESTful API provides endpoints for retrieving exchange rates, creating orders, and tracking order status.

Base URL: `https://0xswap.exchange/api/partner`

API Version: 1.0

Last Updated: January 31, 2026

Table of Contents

Authentication

All API requests require authentication using API keys. You must include both your public and secret keys in the request headers.

Request Headers

```
X-API-Public-Key: your_public_key_here
X-API-Secret-Key: your_secret_key_here
Content-Type: application/json
```

Obtaining API Keys

1. Contact 0xSwap support via https://0xswap.exchange/support (https://0xswap.exchange/support)
2. Request partner API access
3. You will receive:
   - **Public Key**: Used to identify your account (starts with `partner_`)
   - **Secret Key**: Used to authenticate requests (64-character hex string)

> ⚠️ **Security Warning:** Never expose your secret key in client-side code. All API calls MUST be made from your backend server.

Endpoints

1. Get Available Currencies

Retrieve the list of all available cryptocurrencies for exchange.

Endpoint: `GET /api/partner/ccies`

Request Example:

```
curl -X GET https://0xswap.exchange/api/partner/ccies \
  -H "X-API-Public-Key: partner_abc123..." \
  -H "X-API-Secret-Key: c105e76650fda1b9..."
```

Response:

```
{
  "code": 0,
  "data": [
    {
      "code": "BTC",
      "coin": "Bitcoin",
      "network": "BTC",
      "name": "Bitcoin",
      "recv": true,
      "send": true,
      "tag": null,
      "logo": "https://ff.io/assets/images/coins/svg/btc.svg",
      "priority": 100
    },
    {
      "code": "USDTERC20",
      "coin": "USDT",
      "network": "ETH",
      "name": "Tether (ERC20)",
      "recv": true,
      "send": true,
      "tag": null,
      "logo": "https://ff.io/assets/images/coins/svg/usdt.svg",
      "priority": 90
    },
    {
      "code": "USDTTRC",
      "coin": "USDT",
      "network": "TRX",
      "name": "Tether (TRC20)",
      "recv": true,
      "send": true,
      "tag": null,
      "logo": "https://ff.io/assets/images/coins/svg/usdttrc.svg",
      "priority": 90
    }
  ]
}
```


Response Fields:

Field	Type	Description
`code`	string	Currency code (e.g., “BTC”, “USDTERC20”) - **USE THIS IN API CALLS**
`coin`	string	Base coin name (e.g., “Bitcoin”, “USDT”)
`network`	string	Blockchain network (e.g., “BTC”, “ETH”, “TRX”)
`name`	string	Full display name
`recv`	boolean	Whether currency can be received (deposited)
`send`	boolean	Whether currency can be sent (withdrawn)
`tag`	string|null	Memo/tag required (e.g., XRP, XLM)
`logo`	string	URL to currency logo image
`priority`	number	Display priority (higher = more prominent)

---

2. Get Exchange Rate

Calculate the exchange rate between two currencies for a specific amount.

Endpoint: `POST /api/partner/price`

Request Example:

```
curl -X POST https://0xswap.exchange/api/partner/price \
  -H "X-API-Public-Key: partner_abc123..." \
  -H "X-API-Secret-Key: c105e76650fda1b9..." \
  -H "Content-Type: application/json" \
  -d '{
    "fromCcy": "USDTERC20",
    "toCcy": "USDTTRC",
    "direction": "from",
    "amount": 100
  }'
```

Request Body:

Field	Type	Required	Description
`fromCcy`	string	✅ Yes	Source currency code (e.g., “USDTERC20”)
`toCcy`	string	✅ Yes	Destination currency code (e.g., “USDTTRC”)
`direction`	string	✅ Yes	`"from"` (fixed input) or `"to"` (fixed output)
`amount`	number	✅ Yes	Amount to exchange

Response:


```
{
  "code": 0,
  "data": {
    "from": {
      "code": "USDTERC20",
      "network": "ETH",
      "coin": "USDT",
      "amount": "100",
      "rate": 0.995,
      "precision": 8,
      "min": "10",
      "max": "135678.42",
      "usd": 100,
      "btc": 0.00120507
    },
    "to": {
      "code": "USDTTRC",
      "network": "TRX",
      "coin": "USDT",
      "amount": "99.179",
      "rate": 0.995,
      "precision": 8,
      "min": "1",
      "max": "134999.709",
      "usd": 99.18
    },
    "errors": []
  }
}
```


Response Fields:

Field	Description
`from.code`	Source currency code
`from.amount`	Source amount (as string)
`from.min`	**Minimum** exchange amount for source currency
`from.max`	**Maximum** exchange amount for source currency
`from.usd`	USD equivalent of source amount
`to.code`	Destination currency code
`to.amount`	Destination amount after exchange (as string)
`to.usd`	USD equivalent of destination amount
`errors`	Array of validation errors (empty if successful)

> ⚠️ **Important:** Always check `from.min` and `from.max` before creating an order!

---

3. Create Exchange Order

Create a new exchange order with the specified parameters.

Endpoint: `POST /api/partner/create-order`

Request Example:

```
curl -X POST https://0xswap.exchange/api/partner/create-order \
  -H "X-API-Public-Key: partner_abc123..." \
  -H "X-API-Secret-Key: c105e76650fda1b9..." \
  -H "Content-Type: application/json" \
  -d '{
    "fromCcy": "USDTERC20",
    "toCcy": "USDTTRC",
    "direction": "from",
    "amount": 100,
    "toAddress": "TXYZPrarV6ahiFkhEK3z6XZHzqKRr5ksj5",
    "email": "user@example.com"
  }'
```

Request Body:

Field	Type	Required	Description
`fromCcy`	string	✅ Yes	Source currency code (e.g., “USDTERC20”)
`toCcy`	string	✅ Yes	Destination currency code (e.g., “USDTTRC”)
`direction`	string	✅ Yes	`"from"` (fixed input) or `"to"` (fixed output)
`amount`	number	✅ Yes	Amount to exchange
`toAddress`	string	✅ Yes	Recipient wallet address
`toTag`	string	⚠️ Conditional	Memo/tag for XRP, XLM, TON, ATOM, etc.
`email`	string	❌ No	Email for order notifications (optional)

Response (Success):


```
{
  "code": 0,
  "data": {
    "orderNumber": "UHWVPB",
    "status": "NEW",
    "from": {
      "code": "USDTERC20",
      "coin": "USDT",
      "network": "ETH",
      "amount": "100",
      "address": "0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb",
      "tag": null,
      "txId": null
    },
    "to": {
      "code": "USDTTRC",
      "coin": "USDT",
      "network": "TRX",
      "amount": "99.179",
      "address": "TXYZPrarV6ahiFkhEK3z6XZHzqKRr5ksj5",
      "tag": null,
      "txId": null
    },
    "timeLeft": 900
  }
}
```


Response Fields:

Field	Description
`orderNumber`	Unique 6-character order ID (save this!)
`status`	Order status (see Order Statuses)
`from.address`	**Deposit address** - user must send funds here
`from.amount`	Amount user needs to send
`to.address`	Recipient address (where funds will be sent)
`to.amount`	Amount user will receive
`timeLeft`	Seconds remaining to make deposit (typically 900s = 15min)

> ⚠️ **Critical:** User must send exactly `from.amount` to `from.address` within `timeLeft` seconds!

Response (Error - Amount Out of Limits):

```
{
  "code": 1,
  "error": "Amount is below minimum limit. Minimum: 10 USDTERC20"
}
```

---

4. Get Order Status

Track the status of an existing order.

Endpoint: `GET /api/partner/order/{orderNumber}`

Request Example:

```
curl -X GET https://0xswap.exchange/api/partner/order/UHWVPB \
  -H "X-API-Public-Key: partner_abc123..." \
  -H "X-API-Secret-Key: c105e76650fda1b9..."
```

Response:

```
{
  "code": 0,
  "data": {
    "orderNumber": "UHWVPB",
    "status": "PENDING",
    "from": {
      "code": "USDTERC20",
      "coin": "USDT",
      "network": "ETH",
      "amount": "100",
      "address": "0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb",
      "tag": null,
      "txId": "0xabc123..."
    },
    "to": {
      "code": "USDTTRC",
      "coin": "USDT",
      "network": "TRX",
      "amount": "99.179",
      "address": "TXYZPrarV6ahiFkhEK3z6XZHzqKRr5ksj5",
      "tag": null,
      "txId": null
    },
    "timeLeft": 650
  }
}
```


Order Statuses:

Status	Description
`NEW`	Order created, waiting for deposit
`PENDING`	Deposit received, waiting for confirmations
`EXCHANGE`	Funds being exchanged
`DONE`	Exchange completed, funds sent to recipient
`EXPIRED`	Order expired (no deposit within time limit)
`EMERGENCY`	Issue detected, contact support

Error Codes

All API responses follow this structure:

```
{
  "code": 0,
  "data": { ... }    // Present on success (code = 0)
}
```

```
{
  "code": 1,
  "error": "Error message"  // Present on error (code > 0)
}
```

Error Code Reference:

Code	HTTP Status	Description
`0`	200	Success
`1`	400	Bad Request (invalid parameters)
`2`	401	Unauthorized (invalid API keys)
`3`	404	Not Found (order doesn't exist)
`4`	429	Rate Limit Exceeded
`5`	500	Internal Server Error

Common Error Messages:

```
// Missing API credentials
{
  "code": 2,
  "error": "Missing API credentials"
}

// Invalid API keys
{
  "code": 2,
  "error": "Invalid API credentials"
}

// Currency not available
{
  "code": 1,
  "error": "One or both currencies are not available"
}

// Amount out of limits
{
  "code": 1,
  "error": "Amount is below minimum limit. Minimum: 10 USDTERC20"
}

// Order not found
{
  "code": 3,
  "error": "Order not found"
}
```

Currency Codes

Important Notes

1. **USDT has multiple variants** based on blockchain network:
   - `USDTERC20` - USDT on Ethereum (ERC20)
   - `USDTTRC` - USDT on Tron (TRC20)
   - `USDTBSC` - USDT on BSC (BEP20)
   - `USDTMATIC` - USDT on Polygon
   - `USDTARBITRUM` - USDT on Arbitrum
   - `USDTSOL` - USDT on Solana

2. **Always use the FULL code** returned by `/api/partner/ccies`

3. **Check minimum amounts** before creating orders:
   - BTC: typically 0.0005 BTC
   - ETH: typically 0.01 ETH
   - USDTERC20: typically 10 USDT
   - USDTTRC: typically 1 USDT

Popular Currency Pairs

```
USDTERC20 → USDTTRC   (ERC20 to TRC20)
BTC → ETH
ETH → BTC
BTC → USDTTRC
```

Rate Limiting

- **Rate Limit:** 60 requests per minute per API key
- **Burst:** Up to 10 requests per second

Rate Limit Headers:

```
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 45
X-RateLimit-Reset: 1706745600
```

If you exceed the rate limit, you'll receive:

```
{
  "code": 4,
  "error": "Rate limit exceeded. Try again in 30 seconds."
}
```

Best Practices

1. Always Check Rates Before Creating Orders

```
// Step 1: Get rate
const rateResponse = await fetch('https://0xswap.exchange/api/partner/price', {
  method: 'POST',
  headers: {
    'X-API-Public-Key': publicKey,
    'X-API-Secret-Key': secretKey,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    fromCcy: 'USDTERC20',
    toCcy: 'USDTTRC',
    direction: 'from',
    amount: 100
  })
});

const rate = await rateResponse.json();

// Step 2: Check minimums/maximums
if (rate.code === 0) {
  const min = parseFloat(rate.data.from.min);
  const max = parseFloat(rate.data.from.max);
  const amount = 100;

  if (amount < min) {
    console.error(`Amount too low. Minimum: ${min}`);
    return;
  }

  if (amount > max) {
    console.error(`Amount too high. Maximum: ${max}`);
    return;
  }

  // Step 3: Create order
  const orderResponse = await fetch('https://0xswap.exchange/api/partner/create-or-
der', {
    method: 'POST',
    headers: {
      'X-API-Public-Key': publicKey,
      'X-API-Secret-Key': secretKey,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      fromCcy: 'USDTERC20',
      toCcy: 'USDTTRC',
      direction: 'from',
      amount: 100,
      toAddress: 'TXYZPrarV6ahiFkhEK3z6XZHzqKRr5ksj5'
    })
  });

  const order = await orderResponse.json();
  console.log('Order created:', order.data.orderNumber);
}
```

2. Handle Errors Gracefully

```
try {
  const response = await fetch(url, options);
  const data = await response.json();

  if (data.code === 0) {
    // Success
    return data.data;
  } else {
    // API error
    throw new Error(data.error || 'Unknown error');
  }
} catch (error) {
  // Network error or parsing error
  console.error('API Error:', error.message);
  throw error;
}
```

3. Poll Order Status

```
async function waitForOrderCompletion(orderNumber) {
  const maxAttempts = 60; // 5 minutes (5s interval)

  for (let i = 0; i < maxAttempts; i++) {
    const response = await fetch(
      `https://0xswap.exchange/api/partner/order/${orderNumber}`,
      {
        headers: {
          'X-API-Public-Key': publicKey,
          'X-API-Secret-Key': secretKey
        }
      }
    );

    const data = await response.json();

    if (data.code === 0) {
      const status = data.data.status;

      console.log(`Order ${orderNumber} status: ${status}`);

      if (status === 'DONE') {
        return data.data; // Order completed!
      }

      if (status === 'EXPIRED' || status === 'EMERGENCY') {
        throw new Error(`Order ${status}`);
      }

      // Wait 5 seconds before next check
      await new Promise(resolve => setTimeout(resolve, 5000));
    } else {
      throw new Error(data.error);
    }
  }

  throw new Error('Order timeout');
}
```

4. Validate Addresses Before Creating Orders

Always validate wallet addresses on your backend before calling the API:

```
function isValidTronAddress(address) {
  return /^T[A-Za-z1-9]{33}$/.test(address);
}

function isValidEthAddress(address) {
  return /^0x[a-fA-F0-9]{40}$/.test(address);
}

function isValidBtcAddress(address) {
  return /^[13][a-km-zA-HJ-NP-Z1-9]{25,34}$|^bc1[a-z0-9]{39,59}$/.test(address);
}
```

5. Store Order Numbers

Always save `orderNumber` in your database:

```
const order = await createOrder(...);

if (order.code === 0) {
  await db.orders.insert({
    orderNumber: order.data.orderNumber,
    userId: userId,
    fromCcy: 'USDTERC20',
    toCcy: 'USDTTRC',
    amount: 100,
    status: 'NEW',
    createdAt: new Date()
  });
}
```

Example: Complete Integration

```javascript
const OXSWAP_API = 'https://0xswap.exchange/api/partner';
const PUBLIC_KEY = 'partner_abc123...';
const SECRET_KEY = 'c105e76650fda1b9...';

class OxSwapClient {
  async request(endpoint, method = 'GET', body = null) {
    const options = {
      method,
      headers: {
        'X-API-Public-Key': PUBLIC_KEY,
        'X-API-Secret-Key': SECRET_KEY,
        'Content-Type': 'application/json'
      }
    };

    if (body) {
      options.body = JSON.stringify(body);
    }

    const response = await fetch(`${OXSWAP_API}${endpoint}`, options);
    const data = await response.json();

    if (data.code !== 0) {
      throw new Error(data.error || 'API Error');
    }

    return data.data;
  }

  async getCurrencies() {
    return this.request('/ccies');
  }

  async getPrice(fromCcy, toCcy, amount, direction = 'from') {
    return this.request('/price', 'POST', {
      fromCcy,
      toCcy,
      direction,
      amount
    });
  }

  async createOrder(fromCcy, toCcy, amount, toAddress, email = null) {
    return this.request('/create-order', 'POST', {
      fromCcy,
      toCcy,
      direction: 'from',
      amount,
      toAddress,
      email
    });
  }

  async getOrder(orderNumber) {
    return this.request(`/order/${orderNumber}`);
  }
}

// Usage
const client = new OxSwapClient();

// 1. Get available currencies
```

```
const currencies = await client.getCurrencies();
console.log('Available:', currencies.map(c => c.code));

// 2. Get exchange rate
const rate = await client.getPrice('USDTERC20', 'USDTTRC', 100);
console.log(`Rate: ${rate.from.amount} → ${rate.to.amount}`);
console.log(`Min: ${rate.from.min}, Max: ${rate.from.max}`);

// 3. Create order
const order = await client.createOrder(
  'USDTERC20',
  'USDTTRC',
  100,
  'TXYZPrarV6ahiFkhEK3z6XZHzqKRr5ksj5'
);

console.log('Order created:', order.orderNumber);
console.log('Send', order.from.amount, 'to', order.from.address);

// 4. Track order
setInterval(async () => {
  const status = await client.getOrder(order.orderNumber);
  console.log('Status:', status.status);

  if (status.status === 'DONE') {
    console.log('Exchange completed!');
    console.log('TxID:', status.to.txId);
    process.exit(0);
  }
}, 5000);
```

Support

For technical support or to report issues:

- **Email:** support@0xswap.exchange
- **Contact Form:** https://0xswap.exchange/support (https://0xswap.exchange/support)
- **API Status:** https://status.0xswap.exchange (https://status.0xswap.exchange) (coming soon)

Changelog

v1.0 (January 31, 2026)

- Initial Partner API release
- Added `/ccies`, `/price`, `/create-order`, `/order/{orderNumber}` endpoints
- Implemented API key authentication
- Added rate limiting (60 req/min)
- Support for 75+ cryptocurrencies
- Real-time order tracking

Last Updated: January 31, 2026
Version: 1.0