Nansen API - Complete Integration Guide

Overview

All **10 Nansen API endpoints** are now fully integrated into the Smart Money Tracker platform. This document provides a complete reference for using these endpoints.

✅ Integrated Endpoints

1. Smart Money Netflows

Endpoint: `/api/nansen/smart-money?action=netflows`

Purpose: Track token accumulation/distribution by Smart Money wallets

Parameters:
- `action=netflows` (required)
- `chain` - Blockchain (ethereum, base, bnb, polygon, arbitrum, optimism, solana)
- `timeframe` - Time period (1h, 24h, 7d, 30d)
- `limit` - Number of results (default: 50)

Example:

```
GET /api/nansen/smart-money?action=netflows&chain=ethereum&timeframe=24h&limit=100
```

Response:

```
{
  "success": true,
  "action": "netflows",
  "chain": "ethereum",
  "timeframe": "24h",
  "count": 100,
  "data": [
    {
      "tokenAddress": "0x...",
      "tokenSymbol": "ETH",
      "tokenName": "Ethereum",
      "netflow": 1234567.89,
      "inflow": 2000000.00,
      "outflow": 765432.11,
      "smartMoneyCount": 42,
      "signal": "STRONG_BUY"
    }
  ]
}
```

---

2. Smart Money Holdings

Endpoint: `/api/nansen/smart-money?action=holdings`

Purpose: View current token holdings by Smart Money wallets

Parameters:
- `action=holdings` (required)
- `chain` - Blockchain (ethereum, base, bnb, etc.)
- `limit` - Number of results (default: 50)

Example:

```
GET /api/nansen/smart-money?action=holdings&chain=ethereum&limit=50
```

Response:


```
{
  "success": true,
  "action": "holdings",
  "chain": "ethereum",
  "count": 50,
  "data": [
    {
      "tokenAddress": "0x...",
      "tokenSymbol": "LINK",
      "tokenName": "Chainlink",
      "totalHolders": 1250,
      "totalValue": "15234567.89",
      "averageHolding": "12187.65",
      "topHolders": []
    }
  ]
}
```


---

3. Smart Money Historical Holdings ⭐ NEW

Endpoint: `/api/nansen/smart-money?action=historical-holdings`

Purpose: Track how Smart Money positions changed over time

Parameters:
- `action=historical-holdings` (required)
- `tokenAddress` - Token contract address (required)
- `chain` - Blockchain (ethereum, base, bnb, etc.)
- `startDate` - Start date (ISO 8601 format, optional)
- `endDate` - End date (ISO 8601 format, optional)
- `limit` - Number of data points (default: 100)

Example:

```
GET /api/nansen/smart-money?action=historical-holdings&tokenAddress=0x...&chain=eth-
ereum&startDate=2024-01-01&endDate=2024-12-31
```

Response:

```
{
  "success": true,
  "action": "historical-holdings",
  "chain": "ethereum",
  "count": 100,
  "data": [
    {
      "tokenAddress": "0x...",
      "tokenSymbol": "UNI",
      "tokenName": "Uniswap",
      "chain": "ethereum",
      "timestamp": "2024-11-19T12:00:00Z",
      "date": "2024-11-19",
      "totalHolders": 985,
      "totalValue": "8765432.10",
      "totalValueUSD": 8765432.10,
      "averageHolding": "8900.45",
      "medianHolding": "5200.00",
      "changePercent24h": 2.5,
      "topHolders": [
        {
          "walletAddress": "0x...",
          "walletLabel": "Smart DEX Trader",
          "balance": "250000",
          "valueUSD": 250000,
          "percentage": 2.85
        }
      ]
    }
  ]
}
```


---

4. Smart Money DEX Trades

Endpoint: `/api/nansen/smart-money?action=dex-trades`

Purpose: Real-time DEX trading activity from Smart Money wallets

Parameters:
- `action=dex-trades` (required)
- `chain` - Blockchain
- `timeframe` - Time period (1h, 24h, 7d, 30d)
- `limit` - Number of trades (default: 100)

Example:

```
GET /api/nansen/smart-money?action=dex-trades&chain=ethereum&timeframe=1h&limit=50
```

Response:

```
{
  "success": true,
  "action": "dex-trades",
  "chain": "ethereum",
  "timeframe": "1h",
  "count": 50,
  "data": [
    {
      "timestamp": "2024-11-19T12:34:56Z",
      "walletAddress": "0x...",
      "walletLabel": "30D Smart Trader",
      "tokenAddress": "0x...",
      "tokenSymbol": "AAVE",
      "type": "BUY",
      "amountUsd": 50000,
      "amount": 1000,
      "priceUsd": 50,
      "dex": "Uniswap V3",
      "txHash": "0x..."
    }
  ]
}
```


---

5. Token Screener

Endpoint: Already integrated in `nansen-client.ts`

Purpose: Filter and discover tokens based on Smart Money activity

Function: `tokenScreener(filters)`

Example:

```
import { tokenScreener } from '@/lib/nansen-client';

const results = await tokenScreener({
  chain: 'ethereum',
  minSmartMoneyCount: 10,
  minNetflow: 100000,
  timeframe: '24h',
  limit: 50
});
```

---

6. Token God Mode - Flow Intelligence

Endpoint: `/api/nansen/token-intelligence`

Purpose: Analyze token flows across different wallet categories

Parameters:
- `tokenAddress` - Token contract address (required)
- `chain` - Blockchain
- `timeframe` - Time period

Example:

```
GET /api/nansen/token-intelligence?tokenAddress=0x...&chain=ethereum&timeframe=24h
```

Response:

```
{
  "success": true,
  "data": {
    "tokenAddress": "0x...",
    "tokenSymbol": "LINK",
    "chain": "ethereum",
    "smartMoney": {
      "netflow": 1234567,
      "inflow": 2000000,
      "outflow": 765433
    },
    "exchanges": {
      "netflow": -500000,
      "inflow": 300000,
      "outflow": 800000
    },
    "whales": {
      "netflow": 789000,
      "inflow": 1500000,
      "outflow": 711000
    }
  }
}
```

---

7. Token God Mode - Holders

Endpoint: Already integrated in `nansen-client.ts`

Purpose: View top token holders with labels

Function: `getTokenHolders(tokenAddress, chain, limit)`

Example:

```
import { getTokenHolders } from '@/lib/nansen-client';

const holders = await getTokenHolders('0x...', 'ethereum', 100);
```

---

8. Profiler - Address Current Balances

Endpoint: `/api/nansen/wallet-profiler?section=balance`

Purpose: Get current token balances for a wallet

Parameters:
- `address` - Wallet address (required)

- `chain` - Blockchain
- `section=balance`

Example:

```
GET /api/nansen/wallet-profiler?address=0x...&chain=ethereum&section=balance
```

Response:


```
{
  "success": true,
  "data": {
    "address": "0x...",
    "chain": "ethereum",
    "balance": {
      "tokens": [
        {
          "tokenAddress": "0x...",
          "tokenSymbol": "ETH",
          "balance": "10.5",
          "balanceUSD": 52500,
          "price": 5000
        }
      ],
      "totalValueUSD": 52500
    }
  }
}
```


---

9. Profiler - Address Perpetual Positions ⭐ NEW

Endpoint: `/api/nansen/wallet-profiler?section=perp-positions`

Purpose: View open perpetual trading positions

Parameters:
- `address` - Wallet address (required)
- `chain` - Blockchain
- `section=perp-positions`

Example:

```
GET /api/nansen/wallet-profiler?address=0x...&chain=ethereum&section=perp-positions
```

Response:

```
{
  "success": true,
  "data": {
    "address": "0x...",
    "chain": "ethereum",
    "perpPositions": {
      "positions": [
        {
          "protocol": "GMX",
          "protocolName": "GMX Protocol",
          "chain": "arbitrum",
          "tokenAddress": "0x...",
          "tokenSymbol": "BTC",
          "positionType": "LONG",
          "size": 1.5,
          "sizeUSD": 150000,
          "entryPrice": 95000,
          "currentPrice": 100000,
          "leverage": 10,
          "liquidationPrice": 90500,
          "unrealizedPnL": 7500,
          "unrealizedPnLPercent": 5.0,
          "collateral": 15000,
          "collateralUSD": 15000,
          "collateralToken": "USDC",
          "openedAt": "2024-11-15T10:00:00Z",
          "lastUpdated": "2024-11-19T12:00:00Z"
        }
      ],
      "totalPositionValueUSD": 150000,
      "totalUnrealizedPnL": 7500,
      "totalUnrealizedPnLPercent": 5.0,
      "protocols": ["GMX", "dYdX"]
    }
  }
}
```


---

10. Profiler - Address Transactions

Endpoint: `/api/nansen/wallet-profiler?section=transactions`

Purpose: Get wallet transaction history

Parameters:
- `address` - Wallet address (required)
- `chain` - Blockchain
- `section=transactions`
- `limit` - Number of transactions (default: 50)

Example:

```
GET /api/nansen/wallet-profiler?address=0x...&chain=ethereum&section=transactions&limit=100
```

Response:

```
{
  "success": true,
  "data": {
    "address": "0x...",
    "chain": "ethereum",
    "transactions": [
      {
        "txHash": "0x...",
        "timestamp": "2024-11-19T12:00:00Z",
        "from": "0x...",
        "to": "0x...",
        "type": "TOKEN_TRANSFER",
        "tokenAddress": "0x...",
        "tokenSymbol": "USDT",
        "amount": "1000",
        "amountUSD": 1000,
        "gasUsed": 21000,
        "gasPriceGwei": 50
      }
    ]
  }
}
```

Page Integration

Where Each Endpoint is Used

Endpoint	Page/Component	Purpose
Smart Money Netflows	`/smart-money-tracker`	Main tracker tab
Smart Money Holdings	`/smart-money-tracker`	Holdings tab
Historical Holdings ⭐	Coming soon	Historical analysis
DEX Trades	`/smart-money-tracker`	DEX Trades tab
Token Screener	Backend	Token discovery
Flow Intelligence	`/flow-intelligence`	Token flow analysis
Token Holders	Backend	Holder analysis
Wallet Balance	`/wallet-tracker`	Intelligence tab
Perp Positions ⭐	`/wallet-tracker`	Intelligence tab (new)
Wallet Transactions	`/wallet-tracker`	Transactions tab

API Usage in Components

Smart Money Tracker Example

```
'use client';

import { useState, useEffect } from 'react';

export function SmartMoneyData() {
  const [netflows, setNetflows] = useState([]);
  const [holdings, setHoldings] = useState([]);
  const [historicalData, setHistoricalData] = useState([]);

  // Fetch netflows
  useEffect(() => {
    fetch('/api/nansen/smart-money?action=netflows&chain=ethereum&timeframe=24h')
      .then(res => res.json())
      .then(data => setNetflows(data.data));
  }, []);

  // Fetch holdings
  useEffect(() => {
    fetch('/api/nansen/smart-money?action=holdings&chain=ethereum')
      .then(res => res.json())
      .then(data => setHoldings(data.data));
  }, []);

  // Fetch historical holdings for a specific token
  const fetchHistoricalHoldings = (tokenAddress: string) => {
    fetch(`/api/nansen/smart-money?action=historical-holdings&tokenAddress=${tokenAd-
dress}&chain=ethereum`)
      .then(res => res.json())
      .then(data => setHistoricalData(data.data));
  };

  return (
    <div>
      {/* Display netflows, holdings, and historical data */}
    </div>
  );
}
```

Wallet Profiler Example

```
'use client';

import { useState } from 'react';

export function WalletProfiler({ address }: { address: string }) {
  const [profileData, setProfileData] = useState(null);
  const [perpPositions, setPerpPositions] = useState(null);

  // Fetch all wallet profile data
  const fetchProfile = async () => {
    const res = await fetch(
      `/api/nansen/wallet-profiler?address=${address}&chain=ethereum&section=all`
    );
    const data = await res.json();
    setProfileData(data.data);
  };

  // Fetch only perp positions
  const fetchPerpPositions = async () => {
    const res = await fetch(
      `/api/nansen/wallet-profiler?address=${address}&chain=ethereum&section=perp-pos-
itions`
    );
    const data = await res.json();
    setPerpPositions(data.data.perpPositions);
  };

  return (
    <div>
      {/* Display profile data and perp positions */}
    </div>
  );
}
```

Client Library Usage

You can also use the Nansen client directly in server-side code:

```
import {
  getSmartMoneyNetflows,
  getSmartMoneyHoldings,
  getSmartMoneyHistoricalHoldings,
  getSmartMoneyDexTrades,
  getFlowIntelligence,
  getWalletBalance,
  getWalletPerpPositions,
  getWalletTransactions,
} from '@/lib/nansen-client';

// Server component or API route
export async function generateStaticParams() {
  const netflows = await getSmartMoneyNetflows('ethereum', '24h', 50);
  const holdings = await getSmartMoneyHoldings('ethereum', 100);
  const historicalData = await getSmartMoneyHistoricalHoldings(
    '0x...',
    'ethereum',
    '2024-01-01',
    '2024-12-31'
  );
  const perpPositions = await getWalletPerpPositions('0x...', 'ethereum');

  return { netflows, holdings, historicalData, perpPositions };
}
```

Environment Configuration

Make sure you have the Nansen API key configured:

```
# .env
NANSEN_API_KEY=your_nansen_api_key_here
```

Supported Chains

All endpoints support these blockchains:

- ✅ Ethereum (`ethereum`)
- ✅ Base (`base`)
- ✅ BNB Chain (`bnb`)
- ✅ Polygon (`polygon`)
- ✅ Arbitrum (`arbitrum`)
- ✅ Optimism (`optimism`)
- ✅ Solana (`solana`)

Rate Limiting & Caching

- **Cache Duration:** 5 minutes for all endpoints

- **Request Timeout:** 30 seconds
- **Nansen API Credits:** Consumed per request (check your Nansen dashboard)

Error Handling

All API routes include comprehensive error handling:

```
try {
  const data = await getSmartMoneyNetflows('ethereum', '24h', 50);
  // Handle success
} catch (error) {
  // Handle error
  console.error('Nansen API error:', error.message);
}
```

Testing Endpoints

Test Smart Money Netflows

```
curl "https://defidashtracker.com/api/nansen/smart-money?ac-
tion=netflows&chain=ethereum&timeframe=24h&limit=10"
```

Test Historical Holdings

```
curl "https://defidashtracker.com/api/nansen/smart-money?action=historical-hold-
ings&tokenAddress=0x...&chain=ethereum"
```

Test Perp Positions

```
curl "https://defidashtracker.com/api/nansen/wallet-profiler?ad-
dress=0x...&chain=ethereum&section=perp-positions"
```

Test All Wallet Data

```
curl "https://defidashtracker.com/api/nansen/wallet-profiler?ad-
dress=0x...&chain=ethereum&section=all"
```

Summary

✅ **All 10 Nansen API endpoints** are now fully integrated
✅ **2 new endpoints added:** Historical Holdings & Perp Positions
✅ **Complete API routes** for all data types
✅ **Error handling & caching** built-in

✅ **Multi-chain support** across all endpoints
✅ **Ready for production** use

The Smart Money Tracker platform now has comprehensive access to Nansen's institutional-grade blockchain intelligence!

Last Updated: November 19, 2025
Integration Status: Complete ✅