CrymadX User Dashboard

Complete API Documentation for Frontend Developers

Version: 1.0.0 **Base URL:** https://api.crymadx.com/v1 **Last Updated:** December 28, 2025

Table of Contents

80+	17	15
API Endpoints	Screen Modules	WebSocket Events

10

Data Models

1. Overview

The CrymadX API provides programmatic access to the CrymadX cryptocurrency exchange platform.

Request Headers

```
Content-Type: application/json
Accept: application/json
Authorization: Bearer <access_token>
```

Response Format

```
{ "success": true, "data": { ... }, "message": "Operation successful" }
```

Token Types

Token	Expiry	Usage
Access Token	15 min	API requests
Refresh Token	7 days	Get new access token

Rate Limiting

Endpoint	Limit	Window
Public	100	1 min
Authenticated	300	1 min
Trading	50	1 sec
Withdrawals	10	1 min

2. Authentication

POST **`/auth/register`** Public

Creates a new user account.

```
// Request
{ "full_name": "John Doe", "email": "john@example.com", "phone": "+1234567890",
  "password": "SecurePass123!", "confirm_password": "SecurePass123!",
  "referral_code": "REF123", "terms_accepted": true }

// Response (201)
{ "success": true, "data": { "user": { "id": "usr_abc123", "email": "john@example.com" },
  "access_token": "eyJ...", "refresh_token": "eyJ...", "expires_in": 900 }}
```

POST **`/auth/login`** Public

Authenticate user with credentials.

```
// Request
{ "email": "john@example.com", "password": "SecurePass123!", "remember_me": true }

// Response
{ "success": true, "data": { "user": {...}, "access_token": "eyJ...", "requires_2fa": false }}
```

POST **`/auth/login/2fa`** Public

Complete login with 2FA code.

POST **`/auth/refresh`** Public

Refresh access token.

POST **`/auth/logout`** Auth

Logout and invalidate tokens.

POST **`/auth/forgot-password`** Public

Request password reset email.

POST `/auth/reset-password` Public

Reset password with token.

POST `/auth/verify-email` Public

Verify email with code.

POST `/auth/google` Public

Login/Register with Google OAuth.

POST `/auth/apple` Public

Login/Register with Apple OAuth.

3. User / Profile

GET `/user/profile` Auth

Get current user profile.

```
// Response
{ "id": "usr_abc123", "full_name": "John Doe", "email": "john@example.com",
  "kyc_status": "verified", "kyc_level": 2, "two_factor_enabled": true }
```

PUT `/user/profile` Auth

Update user profile.

POST `/user/avatar` Auth

Upload profile avatar (multipart/form-data).

PUT `/user/password` Auth

Change password.

GET `/user/login-history` Auth

Get login history.

POST `/user/2fa/enable` Auth

Enable 2FA authentication.

POST `/user/2fa/verify` Auth

Verify and activate 2FA.

POST `/user/2fa/disable` Auth

Disable 2FA authentication.

GET `/user/api-keys` Auth

Get API keys list.

POST `/user/api-keys` Auth

Create new API key.

DELETE `/user/api-keys/:keyId` Auth

Delete API key.

4. Wallet

GET `/wallet/assets` Auth

Get all wallet assets with balances.

```
// Response
{ "total_balance_usd": 15420.50, "assets": [
  { "symbol": "BTC", "name": "Bitcoin", "spot_available": 0.5,
    "spot_in_order": 0.1, "funding_available": 0.05, "price": 42000.00 }
]}
```

GET `/wallet/deposit/address/:symbol` Auth

Get deposit address for asset.

```
// Response
{ "symbol": "BTC", "network": "Bitcoin", "address": "bc1q...",
  "qr_code": "data:image/png;base64,...", "min_deposit": 0.0001 }
```

POST `/wallet/withdraw` Auth

Request withdrawal.

```
// Request
{ "symbol": "BTC", "network": "Bitcoin", "address": "bc1q...",
  "amount": 0.5, "2fa_code": "123456" }
```

GET `/wallet/withdraw/fee/:symbol` Auth

Get withdrawal fee.

POST `/wallet/convert` Auth

Convert between assets.

GET `/wallet/convert/quote` Auth

Get conversion quote.

POST `/wallet/transfer` Auth

Transfer between spot and funding.

GET `/wallet/transactions` Auth

Get transaction history.

Parameter	Type	Description
type	string	deposit, withdraw, convert, transfer
asset	string	Filter by asset
status	string	pending, completed, failed

5. Trading

GET `/trading/pairs` Public

Get all trading pairs.

GET `/trading/ticker/:pair` Public

Get ticker data.

```
// Response
{ "pair": "BTC/USDT", "last_price": 42150.00, "price_change_24h": 850.00,
  "high_24h": 42500.00, "low_24h": 41200.00, "volume_24h": 15000000.00 }
```

GET `/trading/orderbook/:pair` Public

Get order book.

GET `/trading/trades/:pair` Public

Get recent trades.

POST `/trading/order` Auth

Place a new order.

```
// Request
{ "pair": "BTC/USDT", "side": "buy", "type": "limit", "price": 42000.00, "amount": 0.1 }

// Response
{ "order_id": "ord_abc123", "status": "open", "created_at": "2025-12-28T10:30:00Z" }
```

GET `/trading/orders` Auth

Get open orders.

GET `/trading/orders/history` Auth

Get order history.

DELETE `/trading/orders/:orderId` Auth

Cancel an order.

DELETE `/trading/orders` Auth

Cancel all open orders.

GET `/trading/my-trades` Auth

Get user's trade history.

6. Markets

GET `/markets` Public

Get all market data.

GET `/markets/:symbol` Public

Get specific market data.

GET `/markets/stats` Public

Get market statistics.

GET `/markets/favorites` Auth

Get user's favorite markets.

POST `/markets/favorites/:symbol` Auth

Add to favorites.

DELETE `/markets/favorites/:symbol` Auth

Remove from favorites.

7. P2P Trading

GET `/p2p/orders` Public

Get P2P order listings.

POST `/p2p/orders` Auth

Create P2P order.

POST `/p2p/trade/:orderId` Auth

Initiate P2P trade.

GET `/p2p/trades` Auth

Get user's P2P trades.

POST `/p2p/trades/:tradeId/payment-sent` Auth

Mark payment as sent.

POST `/p2p/trades/:tradeId/payment-received` Auth

Confirm payment received.

POST `/p2p/trades/:tradeId/cancel` Auth

Cancel P2P trade.

POST `/p2p/trades/:tradeId/dispute` Auth

Open dispute on trade.

POST `/p2p/trades/:tradeId/chat` Auth

Send chat message.

8. Earn / Savings

GET `/earn/products` Public

Get all savings products.

POST `/earn/subscribe` Auth

Subscribe to savings product.

GET `/earn/subscriptions` Auth

Get user's subscriptions.

POST `/earn/redeem/:subscriptionId` Auth

Redeem from savings.

GET `/earn/interest-history` Auth

Get interest history.

9. Vault

GET `/vault/options` Public

Get vault lock options.

POST `/vault/create` Auth

Create new vault lock.

GET `/vault/positions` Auth

Get user's vault positions.

POST `/vault/break/:positionId` Auth

Break vault early (with penalty).

POST `/vault/claim/:positionId` Auth

Claim matured vault.

10. Referral

GET `/referral/stats` Auth

Get referral statistics.

GET `/referral/list` Auth

Get referral list.

GET `/referral/payouts` Auth

Get payout history.

POST `/referral/withdraw` Auth

Withdraw referral earnings.

11. Rewards

GET `/rewards/tasks` Auth

Get available reward tasks.

POST `/rewards/claim/:taskId` Auth

Claim task reward.

GET `/rewards/tiers` Public

Get reward tiers.

GET `/rewards/history` Auth

Get reward history.

12. Support / Tickets

GET `/support/categories` Public

Get support categories.

POST `/support/tickets` Auth

Create support ticket.

GET `/support/tickets` Auth

Get user's tickets.

GET `/support/tickets/:ticketId` Auth

Get ticket details.

POST `/support/tickets/:ticketId/reply` Auth

Reply to ticket.

POST `/support/tickets/:ticketId/close` Auth

Close ticket.

13. Notifications

GET `/notifications` Auth

Get user notifications.

PUT `/notifications/:notificationId/read` Auth

Mark as read.

PUT `/notifications/read-all` Auth

Mark all as read.

GET `/notifications/settings` Auth

Get notification settings.

PUT `/notifications/settings` Auth

Update notification settings.

14. WebSocket Events

URL: `wss://api.crymadx.com/ws`

Public Channels

Channel	Event	Description
ticker:{pair}	ticker_update	Real-time price updates
orderbook:{pair}	orderbook_update	Order book changes
trades:{pair}	trade	New trades executed
markets	market_update	All market updates

Private Channels (Auth Required)

Channel	Event	Description
user:{userId}	order_update	Order status changes
user:{userId}	trade_executed	User's trade executed
user:{userId}	balance_update	Balance changes
user:{userId}	deposit_confirmed	Deposit confirmed
user:{userId}	withdrawal_processed	Withdrawal processed
user:{userId}	notification	New notification
p2p:{tradeId}	p2p_message	P2P chat message

15. Data Models

User

```
interface User {
  id: string; full_name: string; email: string; phone: string;
  kyc_status: 'pending' | 'verified' | 'rejected'; kyc_level: 0|1|2|3;
  two_factor_enabled: boolean; email_verified: boolean;
}
```

WalletAsset

```
interface WalletAsset {
  symbol: string; name: string; spot_available: number;
  spot_in_order: number; funding_available: number; price: number;
}
```

Order

```
interface Order {
  id: string; pair: string; side: 'buy'|'sell'; type: 'market'|'limit';
  price: number; amount: number; filled: number; status: string;
}
```

Transaction

```
interface Transaction {
  id: string; type: 'deposit'|'withdraw'|'convert'|'transfer';
  asset: string; amount: number; status: string; tx_hash: string;
}
```

16. Error Codes

Authentication Errors

Code	HTTP	Description
AUTH_INVALID_CREDENTIALS	401	Invalid email or password
AUTH_TOKEN_EXPIRED	401	Access token has expired
AUTH_TOKEN_INVALID	401	Invalid token
AUTH_2FA_REQUIRED	403	2FA verification required

Wallet Errors

Code	HTTP	Description
WALLET_INSUFFICIENT_BALANCE	400	Insufficient funds
WALLET_INVALID_ADDRESS	400	Invalid address
WALLET_MIN_AMOUNT	400	Below minimum

Trading Errors

Code	HTTP	Description
TRADING_PAIR_NOT_FOUND	404	Pair not found
TRADING_ORDER_FAILED	400	Order failed
TRADING_MIN_ORDER	400	Below min order

General Errors

Code	HTTP	Description
RATE_LIMIT_EXCEEDED	429	Too many requests
VALIDATION_ERROR	422	Validation failed
SERVER_ERROR	500	Internal server error

CrymadX User Dashboard API Documentation

Version 1.0.0 | December 28, 2025

For Admin Dashboard API, see `CrymadX_Admin_API_Documentation.html`