Documentation Index

Fetch the complete documentation index at: /llms.txt

Use this file to discover all available pages before exploring further.

Skip to main content
Polymarket Documentation home page

Polymarket

English
Search... ⌘K

- Get Help
- Main Site
- Main Site

Search...
Navigation
Core Concepts
Polymarket USD
Predictions
Perps
API Reference
Changelogs

Getting Started

- Overview
- Polymarket 101
- SDKs & APIs

Core Concepts

Archived copy · cryp2nova.com

- Markets & Events
- Negative Risk Markets
- Prices & Orderbook
- Positions & Tokens
- Polymarket USD
- Order Lifecycle
- Resolution

Market Data

- Overview
- Discover Markets
- Market Details
- Prices and Order Books
- Analytics
- Real-Time Data
- Chainlink TWAP Prices

Trading

- Overview
- Place Your First Order
- Wallets and Authentication
- Place Orders
- Manage Orders
- Real-Time Order Updates
- Positions
- Combos
- Wallet Activity
- Fees
- Market Making
- Matching Engine Restarts
- Bridge

CRYP2NOVA

Archived copy · cryp2nova.com

Polymarket USD

pUSD (Polymarket USD) is the collateral token used for all trading on Polymarket. It's a standard ERC-20 token on Polygon, backed by USDC. The smart contract — which enables the withdrawal functionality — enforces the backing. No algorithmic peg, no fractional reserve.

Day to day, nothing changes. You load funds, see a balance, trade, and withdraw. pUSD is the technical settlement layer underneath the same experience you're used to.

Why pUSD

The protocol settles all trading activity in native USDC, providing a more capital efficient, scalable, and institutionally aligned settlement standard as the platform continues to grow. pUSD is a standard ERC-20 wrapper that represents a USDC claim. Wrapping and unwrapping are enforced onchain by the `CollateralOnramp` and `CollateralOfframp` contracts.

Key facts

Archived copy · cryp2nova.com

Token standard	ERC-20
Network	Polygon mainnet
Decimals	6
Backing	USDC (enforced onchain)
Transferable	Yes — standard ERC-20

pUSD is designed to function within Polymarket. There are no current plans to list it on external exchanges. See the Contracts page for all collateral-related contract addresses.

Wrapping — USDC.e → pUSD

Use the **CollateralOnramp** to wrap USDC.e into pUSD.

```
function wrap(address _asset, address _to, uint256 _amount) external
```

Parameters

- `_asset` — address of the asset being wrapped. Must be USDC.e.
- `_to` — recipient of the minted pUSD. Does not have to be `msg.sender`.
- `_amount` — amount to wrap, in USDC.e base units (6 decimals).

Requirements

- The caller must first approve the **CollateralOnramp** contract (not the pUSD token) to spend USDC.e.
- Reverts with `OnlyUnpaused()` if the admin has paused USDC.e.

Example

TypeScript | Python

```
import {
  createWalletClient,
  createPublicClient,
  http,
  parseAbi,
  parseUnits,
} from "viem";
import { polygon } from "viem/chains";
import { privateKeyToAccount } from "viem/accounts";

const account = privateKeyToAccount(process.env.PRIVATE_KEY as `0x${string}`);
const walletClient = createWalletClient({ account, chain: polygon, transport: http() });
const publicClient = createPublicClient({ chain: polygon, transport: http() });

const ONRAMP = "0x93070a847efEf7F70739046A929D47a521F5B8ee" as const;
const USDCE = "0x2791Bca1f2de4661ED88A30C99A7a9449Aa84174" as const; // USDC.e on Polygon

const amount = parseUnits("100", 6); // 100 USDC.e

// 1. Approve the Onramp to spend your USDC.e
const approveHash = await walletClient.writeContract({
  address: USDCE,
  abi: parseAbi(["function approve(address spender, uint256 amount) returns (bool)"]),
  functionName: "approve",
  args: [ONRAMP, amount],
});
await publicClient.waitForTransactionReceipt({ hash: approveHash });

// 2. Wrap USDC.e → pUSD
const wrapHash = await walletClient.writeContract({
  address: ONRAMP,
  abi: parseAbi(["function wrap(address _asset, address _to, uint256 _amount)"]),
  functionName: "wrap",
  args: [USDCE, account.address, amount],
```

CRYP2NOVA

Archived copy · cryp2nova.com

```
});
await publicClient.waitForTransactionReceipt({ hash: wrapHash });
```

Unwrapping — pUSD → USDC.e

Use the **CollateralOfframp** to unwrap pUSD back into USDC.e.

```
function unwrap(address _asset, address _to, uint256 _amount) external
```

Parameters

- `_asset` — asset you want to receive. Must be USDC.e.
- `_to` — recipient of the underlying asset.
- `_amount` — amount of pUSD to unwrap (6 decimals).

Requirements

- The caller must first approve the **CollateralOfframp** contract to spend their pUSD.
- Same pause gate as the Onramp.

Next steps

↗

Contracts

All Polymarket contract addresses and audits
↗

Bridge

Deposit from other chains — auto-wraps to pUSD

Was this page helpful?

Yes No

Positions & Tokens
Previous
Order Lifecycle
Next

Ask a question... ⌘I

Archived copy · cryp2nova.com

Polymarket



Archived copy · cryp2nova.com