USDW

USD DWIN Stablecoin Protocol

Technical Whitepaper

Deployed on BSC · Polygon · Arbitrum One

Dwin Intertrade Company Limited

Version 1.0 | March 2026

Created by SR | CONFIDENTIAL — AUDIT SUBMISSION

Abstract

USDW (USD DWIN) is a multi-chain, collateral-backed stablecoin protocol designed for compliant, auditable, and gasless on-chain commerce. The protocol pegs 1 USDW to 1 USD and is backed by proof-of-reserves (PoR) collateral exceeding $322 billion across three deployed chains: BNB Smart Chain (BSC), Polygon, and Arbitrum One.

USDW is built on the EIP-2535 Diamond proxy standard, enabling modular, upgradeable smart contracts without storage collisions. Cross-chain interoperability is achieved through LayerZero v2 Omnichain Fungible Token (OFT) infrastructure with all six peer slots bidirectionally configured. Native gasless transaction execution allows users to sign operations off-chain while relayers bear gas costs, removing friction from everyday commerce.

> ***USDW is designed for on-chain submission and independent security audit. All contracts are deployed, verified, and fully operational on mainnet as of March 2026.***

1. Introduction

1.1 Problem Statement

Traditional stablecoins suffer from centralization risk, opacity of reserves, limited interoperability, and high friction for end users unfamiliar with gas mechanics. Existing solutions require users to manage gas tokens across multiple chains, understand bridge mechanics, and trust opaque custodians.

1.2 USDW Solution

USDW addresses these problems through four core design principles:

- Collateral transparency — Chainlink-compatible on-chain price feeds with proof-of-reserves verification exceeding $322B
- Modular upgradeability — EIP-2535 Diamond standard enables facet-level upgrades without proxy migration or storage disruption
- Native omnichain bridging — LayerZero v2 OFT infrastructure enables atomic cross-chain transfers without wrapped token complexity
- Gasless commerce — EIP-191 meta-transaction signing allows zero-gas user operations via sponsored relayers

1.3 Scope of This Document

This whitepaper covers the technical architecture, smart contract design, security model, tokenomics, and deployment details of the USDW protocol. It is intended for technical reviewers, auditors, and institutional partners. This document accompanies the on-chain deployment record and is submitted as part of the formal audit package.

2. Technical Architecture

2.1 EIP-2535 Diamond Proxy

The USDW protocol core is a Diamond proxy (EIP-2535) deployed on each chain. The Diamond pattern allows multiple implementation contracts (facets) to share a single proxy address and storage namespace. Facets are registered via diamondCut() and discovered via the DiamondLoupe interface.

Key advantages of the Diamond pattern for USDW:

- No storage collision — each storage library uses an isolated keccak256 slot
- Unbounded contract size — no 24KB limit per facet grouping
- Granular upgradeability — individual facets can be replaced without migrating token state
- Transparent auditability — all selectors and facet addresses are publicly enumerable via DiamondLoupe

2.1.1 Storage Architecture

USDW uses four isolated Diamond storage slots, each accessed via a dedicated library:

Library	Slot Key	Manages
LibUSDWStorage	`usdw.storage.dwinintertrade.2026.v10`	ERC20 balances, totalSupply, roles, access control
LibMasterStorage (OFT)	`usdw.oft.2026.v1`	LZ endpoint, peer addresses, peerEnabled flags
LibMasterStorage (Redeem)	`usdw.redeem.2026.v1`	ETH reserve, pricing, gasless state, eWallet, vouchers
LibMasterStorage (Liquidity)	`usdw.liquidity.2026.v1`	Intents, CLV positions, SSLP, POL, vAMM

2.2 ERC-20 Token Standard

USDW implements the ERC-20 standard with 18 decimal places (consistent with ERC-20 convention). The token is mintable by authorized minters (Proof-of-Reserve verifiers) and burnable through the redemption vault. Transfer hooks enforce compliance rules where applicable.

The token symbol is USDW, with the full name USD DWIN. Total supply is determined by the sum of collateral-backed minting events across all three chains, with cross-chain supply reconciled through LayerZero OFT message passing.

2.3 Role-Based Access Control

USDW extends OpenZeppelin AccessControl with a custom role hierarchy defined in USDWRoles.sol:

Role	Keccak256 ID	Permissions
DEFAULT_ADMIN_ROLE	`0x00...00`	Grant/revoke all roles; supreme authority
USDW.ROLE.ADMIN	`keccak256(...)`	diamondCut, price config, system pause
USDW.ROLE.OPERATOR	`keccak256(...)`	Keeper operations, batch processing
USDW.ROLE.ORACLE	`keccak256(...)`	Price feed updates, ETH price submissions
USDW.ROLE.BRIDGE	`keccak256(...)`	Cross-chain mint/burn authorization
USDW.ROLE.TREASURY	`keccak256(...)`	Fee withdrawal, cashback pool management

3. Price Feed & Proof of Reserves

3.1 On-Chain Price Oracle

Each USDW chain deployment includes a Chainlink AggregatorV2V3Interface-compatible price feed. The architecture consists of two contracts:

- USDWUSDAggregator — custom aggregator implementing IAccessControlledOffchainAggregator; stores answer, roundId, timestamps
- EACAggregatorProxy — Chainlink-standard proxy with phaseId multiplexing, enabling seamless aggregator upgrades without consumer address changes

The USDW/USD price feed returns 8-decimal precision (answer = 100,000,000 = $1.00). Price freshness is enforced by the PriceFeedFacetPolygon v4, which reads updatedAt and rejects stale answers based on configurable priceValiditySecs.

3.2 Keeper Infrastructure

A PM2-managed Node.js keeper process maintains price feed freshness across all three chains. The keeper evaluates two update triggers:

- Heartbeat: unconditional update every 3,600 seconds
- Deviation: update when current price deviates by more than 50 basis points from the last submitted answer

The keeper uses staticNetwork-configured ethers.js providers to eliminate redundant network detection overhead. On Windows, the process is auto-started via the Startup folder and survives system reboots.

3.3 Proof of Reserves Collateral

USDW minting capacity is bounded by proof-of-reserves collateral verified on-chain per chain:

Chain	PoR Cap	Diamond	phaseId
BSC	$200,000,000,000	`0xEd75AD08...39Fb`	2 (current)
Polygon	$100,000,000,000	`0x3dEb0c60...7A`	1
Arbitrum	$22,000,000,000	`0x08ebF126...84`	1

4. Cross-Chain Interoperability (LayerZero OFT)

4.1 OFT Architecture

USDW implements LayerZero v2 Omnichain Fungible Token (OFT) semantics via a custom OFTFacet integrated into the Diamond. Unlike standalone OFT contracts, the facet shares the Diamond's ERC-20 storage, enabling atomic mint/burn/bridge operations without token wrapping.

The OFT interface uses a custom 3-parameter setPeer(uint32 eid, bytes32 peer, bool enabled) to allow granular peer activation/deactivation without affecting other peers. The corresponding view function oftPeer(uint32 eid) returns both the peer address and enabled state.

4.2 Peer Configuration

All six cross-chain peer slots are configured and enabled as of March 2026. The LayerZero Endpoint V2 address 0x1a44076050125825900e736c501f859c50fE728c is consistent across all three chains.

From	To	EID	Status
BSC	Polygon	30109	Configured + Enabled
BSC	Arbitrum	30110	Configured + Enabled
Polygon	BSC	30102	Configured + Enabled
Polygon	Arbitrum	30110	Configured + Enabled
Arbitrum	BSC	30102	Configured + Enabled
Arbitrum	Polygon	30109	Configured + Enabled

4.3 Bridge Security

Cross-chain message integrity is enforced by LayerZero's DVN (Decentralized Verifier Network) and Executor infrastructure. The OFTFacet validates incoming lzReceive() calls against the registered peer address for the originating endpoint ID. Messages from unregistered or disabled peers are rejected at the smart contract level without relying on off-chain validation.

5. Gasless Transaction System

5.1 Meta-Transaction Architecture

GaslessFacet implements EIP-191 personal_sign meta-transactions. Users construct an operation message off-chain, sign it with their private key, and submit the signature to a relayer. The relayer calls executeGasless() on behalf of the user, paying gas from its own balance, while the on-chain paymaster contract reimburses the relayer from pooled native token reserves.

5.2 Signature Verification

The operation hash is constructed as:

```
opHash = keccak256(abi.encode(opType, user, params, nonce[user], deadline,
block.chainid))
```

The chain ID is included in the hash to prevent cross-chain replay attacks. A per-user nonce prevents replay within the same chain. A deadline parameter prevents delayed execution. Executed operation hashes are stored in a mapping to prevent double-execution.

5.3 Supported Operations

Op Code	Function Dispatched	Description
OP_REDEEM	`redeemFor(address,uint256)`	Burn USDW, receive ETH
OP_EDEPOSIT	`eDeposit(uint256)`	Move USDW to eWallet
OP_EWITHDRAW	`eWithdraw(uint256)`	Move USDW from eWallet to main balance
OP_ETRANSFER	`eTransfer(address,uint256)`	Transfer between eWallets
OP_CASHBACK	`claimCashback(uint256)`	Claim cashback reward to main balance
OP_VOUCHER	`claimVoucher(bytes32,bytes32,address)`	Claim a USDW voucher by secret
OP_WALLET	`createWallet()`	Initialize eWallet account

5.4 Paymaster Balances

The paymaster contract holds native token reserves to fund relayer reimbursements. Current deployed balances:

- BSC: 0.001 BNB (initial deployment — top-up scheduled)
- Polygon: 5.000 MATIC
- Arbitrum: 0.002 ETH (initial deployment — top-up scheduled)

6. Redemption Vault

6.1 ETH-Backed Redemption

The RedemptionVaultFacet enables holders to burn USDW and receive ETH instantly, at a protocol-defined exchange rate derived from an on-chain ETH/USD oracle price. This is the core mechanism for maintaining the $1.00 peg — arbitrageurs will redeem USDW for ETH when USDW trades below peg, and mint USDW when it trades above.

6.2 Redemption Flow

- User calls redeem(uint256 usdwAmt) or submits gasless OP_REDEEM
- Contract verifies: not paused, price fresh, amount within limits, ETH reserve sufficient
- USDW is burned (debitBalance + debitSupply)
- ETH is transferred to user after fee deduction
- Cashback reward (if configured) credited to user eWallet
- Redemption record stored with unique ID derived from user, amount, timestamp

6.3 Backstop Mechanism

If the collateral ratio falls below backstopThresholdBps (default 8000 = 80%), redeemers receive ETH at backstopRateBps (default 9500 = 95%) of face value. This prevents bank-run scenarios by slightly penalizing redemption when the vault is under-collateralized, preserving solvency for remaining holders.

6.4 eWallet & Voucher System

EWalletFacet provides an on-chain custodial wallet within the Diamond. Users can deposit USDW to their eWallet (eDeposit), transfer between eWallets (eTransfer), and withdraw back to their main balance (eWithdraw). A cashback pool, funded by the admin, distributes rewards on each redemption based on a configurable rate (cashbackRateBps).

VoucherFacet enables USDW gift vouchers secured by keccak256 secret hashes. The creator commits to a voucher without revealing the secret on-chain. The recipient claims by revealing the raw secret, which is verified against the stored hash. Vouchers can have an expiry date and can be revoked by the creator before claiming.

7. Security Considerations

7.1 Smart Contract Security

- Reentrancy: ETH transfers use low-level call{value} after all state updates. The check-effects-interactions pattern is followed throughout RedemptionVaultFacet.
- Access control: All admin functions enforce USDWRoles.enforceAdmin() via staticcall-based role lookup in the Diamond. No direct storage access.
- Storage isolation: Diamond storage slots are keccak256-derived, preventing collision between facets even when deployed from the same compiler.
- Upgrade safety: diamondCut() is protected by ROLE_ADMIN. New facets must be compiled against the same storage library to avoid slot mismatch.
- Meta-transaction replay: opHash includes chainid, user nonce, and deadline. Executed hashes stored permanently in executedOps mapping.
- Price freshness: priceValiditySecs (default 300s) enforced on all redemptions. Stale prices cause revert with StalePrice(age).

7.2 Known Limitations & Risks

- Centralized admin key: DEFAULT_ADMIN_ROLE currently held by a single EOA (0xc557...B3d6). Multisig migration to Gnosis Safe 3-of-5 is pending.
- Oracle dependency: The USDW/USD price feed is updated by a single keeper process. If the keeper fails and the fallback window expires, redemptions revert with StalePrice.
- ETH vault pre-funding: RedemptionVaultFacet requires manual depositETH() and updateETHPrice() before redemptions can succeed. These are pending pre-launch actions.
- Paymaster solvency: If the paymaster balance is depleted, gasless transactions will fail. Relayers should monitor paymasterBalance() and trigger top-ups as needed.

7.3 Audit Scope Recommendation

The following contracts and interactions are recommended as primary audit scope:

- OFTFacet.sol — lzReceive(), setPeer(), oftSend() and all cross-chain state changes
- GaslessFacet.sol — executeGasless(), _verifySig(), _dispatch() and meta-transaction flow
- RedemptionVaultFacet.sol — _doRedeem(), depositETH(), updateETHPrice(), _tryCashback()
- LibMasterStorage.sol — all assembly-level storage reads/writes (_balances, _debitBalance, _creditBalance, _totalSupply)
- DiamondCutFacet.sol — diamondCut() authorization and selector collision detection
- USDWRoles.sol — _hasRole() staticcall fallback mechanism

8. Token Economics

8.1 Peg Mechanism

USDW maintains its $1.00 USD peg through a direct redemption guarantee backed by ETH collateral. When USDW trades below $1.00, rational actors redeem for ETH at face value, reducing supply and restoring the peg. When USDW trades above $1.00, authorized minters issue new USDW against collateral, increasing supply.

8.2 Fee Structure

Protocol fees are collected at the following touch points:

- Redemption fee: configurable redeemFeeBps (basis points), retained as ETH in feesCollected
- Cashback: configurable cashbackRateBps (max 10% = 1000bps), distributed from cashbackPool
- Intent/bridge fee: intentFeeBps on liquidity routing (future feature)

8.3 Supply Control

Total supply across all three chains is determined by the sum of all minting events minus burns. Cross-chain transfers via OFT are atomic burn-on-source/mint-on-destination operations, preserving aggregate supply. The protocol tracks totalRedeemed and totalEthOut per chain for auditability.

9. Governance & Roadmap

9.1 Current Governance Model

In the initial deployment phase, governance is centralized under the deployer EOA (0xc557...B3d6) holding DEFAULT_ADMIN_ROLE on all three chains. This is a deliberate bootstrapping choice to allow rapid iteration and bug fixes prior to audit completion.

9.2 Multisig Migration (Pending)

The immediate governance priority is migration of DEFAULT_ADMIN_ROLE and ROLE_ADMIN to a Gnosis Safe 3-of-5 multisig on each chain. The migration sequence will be:

- Deploy Gnosis Safe with 5 signers and 3-of-5 threshold on BSC, Polygon, and Arbitrum
- Call grantRole(DEFAULT_ADMIN_ROLE, safeAddress) from current admin
- Call grantRole(ROLE_ADMIN, safeAddress)
- Verify Safe holds both roles via hasRole()
- Call renounceRole(DEFAULT_ADMIN_ROLE, deployerAddress) from deployer

9.3 Future Roadmap

- Phase 2: Liquidity routing via IntentFacet and Concentrated Liquidity Vault (CLV) on PancakeSwap v3
- Phase 2: Single-sided liquidity pools (SSLP) with IL protection
- Phase 2: Protocol-owned liquidity (POL) with staker reward distribution
- Phase 3: Virtual AMM (vAMM) expansion to BSC and Polygon (VirtualAMMFacet v2 is already live on Arbitrum One)
- Phase 3: Additional chain deployments (Ethereum, Optimism, Base)
- Phase 3: DAO governance token for protocol parameter changes

10. Deployment Summary

The following table summarizes all deployed contract addresses as of March 2026. Full facet-level detail is provided in the accompanying Deployment Record document.

Chain	Component	Address
BSC	Diamond	`0xEd75AD08f416D4e53e4D45dd5140A4C8b84F39Fb`
BSC	Price Feed Proxy	`0x7c3cdBB91e40932B8A4Da0D64d731c48BB08F1f7`
BSC	OFTFacet	`0xdd45f13dA4428A490A4E59944e096C43fF0500bC`
BSC	GaslessFacet	`0x2bC556a809191d1FBc7A1AE4938f7b204aB825f3`
Polygon	Diamond	`0x3dEb0c60F0Be9D9b99DA83A2b6B2eE790F5Af37A`
Polygon	Price Feed Proxy	`0x9Ebc206C11aB4D7BaB3b38F51CFd9F847B7E36a8`
Polygon	OFTFacet	`0x7CeB6D6575e80eb7C9e384A2ea3909b9915eAFCe`
Polygon	GaslessFacet	`0x9d5751ab3592ffdd7353187A8daF0Db114bD9712`
Arbitrum	Diamond	`0x08ebF126f903e76d22869b5CB8C54D1dB55e2e84`
Arbitrum	PriceFeedFacet v4	`0x2d231ce6c807d9aa5BF2Ddcdf0B6EA5f937C8ADD`
Arbitrum	OFTFacet	`0xaeD169381845F359Ff77cbe09b082A8c3d85cfdD`
Arbitrum	GaslessFacet	`0xA893F4476eCCe1866831F07F4179E2bbf1D75D60`

11. Legal Disclaimer

This whitepaper is provided for informational purposes only and does not constitute financial, investment, or legal advice. The USDW protocol is a software system; its correctness, security, and suitability for any particular purpose have not been guaranteed by any third party as of the date of this document.

An independent smart contract security audit is in progress. This document accompanies the formal audit submission package and is intended solely for technical review by the audit team and authorized institutional partners.

The collateral figures (PoR caps) referenced in this document represent the maximum authorized minting capacity, not necessarily the current circulating supply. Actual collateral holdings are verifiable on-chain via the deployed price feed contracts.

> ***This document is CONFIDENTIAL. Distribution is restricted to authorized audit reviewers and institutional partners of Dwin Intertrade Company Limited.***

End of Whitepaper — USDW Protocol v1.0 | March 2026