Pandacea Protocol - System Design Document (SDD)

Version:	1.1
Status:	**Draft**
Date:	July 12, 2025
Lead Engineer:	Asa Conant
Related Documents:	Pandacea Technical Whitepaper (v3.1), Pandacea Go-to-Market Plan (v1.1)

1. Overview

1.1. Purpose

This System Design Document (SDD) provides the engineering blueprint for the Pandacea Protocol Minimum Viable Product (MVP). It translates the architectural concepts outlined in the Technical Whitepaper into concrete data models, service interactions, and smart contract interfaces. This document will guide the development team in building a secure, scalable, and robust initial version of the protocol.

1.2. MVP Scope & Objectives

The primary objective of the MVP is to validate the core economic loop of the Pandacea Protocol within the **autonomous warehouse logistics** market.

Key Objectives:

1. Enable **Earners** (e.g., warehouse operators) to list and monetize specific, high-value logistics data.
2. Enable **Spenders** (e.g., robotics AI developers) to discover, lease, and utilize this data in a privacy-preserving manner.
3. Successfully implement and test the foundational economic mechanisms: Heuristic-Based Pricing (PDVF), Dynamic Minimum Pricing (DMP), and Reputation-Based Royalties (RBR).
4. Establish a secure and auditable "Safe Harbor" through verifiable consent on the Polygon PoS network.

2. System Architecture

2.1. Logical Architecture Diagram

The diagram below illustrates the high-level components of the Pandacea Protocol and their primary interactions.

```
graph TD
    subgraph User Layer
        A[MyData Agent <br/>(Earner)]
        B[Buyer-Side Agent <br/>(Spender)]
    end

    subgraph Protocol Layer
        C(P2P Network <br/>- libp2p KAD-DHT)
        D(Storage Layer <br/>- IPFS/Filecoin)
        E(Privacy Layer <br/>- OpenMined PySyft)
    end

    subgraph Settlement Layer (Polygon PoS)
        F[LeaseAgreement Contract]
        G[RoyaltyDistributor Contract]
        H[Reputation Contract]
        I[PGT Token Contract]
    end

    A -- Publishes DataProduct --> C
    B -- Discovers DataProduct --> C
    C -- P2P Communication --> A
    C -- P2P Communication --> B
    A -- Negotiates Storage Deal --> D
    B -- Initiates Lease --> F
    F -- Interacts with --> G
    F -- Interacts with --> H
    F -- Interacts with --> I
    A -- Approves Lease via --> F
    E -- Facilitates Computation --> A
    E -- Returns Result --> B
```

2.2. Component Breakdown

- **MyData Agent (Earner):** A user-controlled application (initially desktop/server

for warehouse context) that manages data sources, enforces user policies (via OPA), and acts as a P2P node for data leasing and privacy-preserving computation.
- **Buyer-Side Agent (Spender):** A developer-focused application for defining data requirements, discovering data products, initiating leases, and receiving the results of data science tasks.
- **P2P Network (libp2p):** The communication backbone for agent discovery, direct negotiation, and data exchange, using a Kademlia DHT for decentralized node lookup.
- **Storage Layer (IPFS/Filecoin):** Decentralized storage for data payloads. All data is content-addressed via IPFS CIDs. For payloads over 10 MB, a Filecoin storage deal is negotiated and verified on-chain to ensure persistent, long-term availability.
- **Privacy Layer (PySyft):** The framework for executing remote data science tasks. For the MVP, this will focus on **federated analysis and statistics on extracted features**, rather than end-to-end training of complex models, to ensure feasibility on typical industrial PC hardware.
- **Settlement Layer (Polygon PoS):** The public blockchain for recording all value-based transactions, enforcing agreements via smart contracts, and distributing royalties.

2.3. Technology Stack (MVP)

- **P2P Networking:** Go-libp2p
- **Blockchain:** Polygon PoS
- **Smart Contracts:** Solidity
- **Storage:** IPFS, Filecoin
- **Agent Backend:** Go / Python
- **Agent Frontend:** Electron / React
- **Privacy Computation:** PySyft

3. Core Data Models & Schemas (MVP Focus)

This section defines the initial data structures for the MVP, focusing on the two prioritized data types.

3.1. DataProduct Schema

This is the public-facing listing that an Earner publishes to the DHT to advertise their available data.

{

```
  "schemaVersion": "1.1",
  "productId": "did:pandacea:earner:123/abc-456",
  "ownerDid": "did:pandacea:earner:123",
  "name": "Novel Package 3D Scans - Warehouse A",
  "description": "High-fidelity point cloud data of irregularly shaped packages
captured by a robotic arm-mounted LiDAR sensor.",
  "dataType": "RoboticSensorData",
  "keywords": ["robotics", "3d-scan", "lidar", "logistics", "grasping"],
  "sampleCid": "bafybeig...xyz" // IPFS CID of a small, representative sample
}
```

3.2. MVP Data Type 1: RoboticSensorData

The schema for the actual data payload referenced by a DataProduct of this type. This metadata is stored alongside the raw data file (e.g., a .pcd file) on IPFS.

```
{
  "schemaVersion": "1.1",
  "captureTimestamp": "2025-07-12T19:50:00Z",
  "sensorType": "LIDAR",
  "sensorModel": "Velodyne Puck VLP-16",
  "dataFormat": "PCD_BINARY",
  "payloadCid": "bafybeig...abc", // IPFS CID of the actual .pcd file
  "storageDealId": "123456", // On-chain Filecoin deal ID if payload > 10MB
  "metadata": {
    "robotId": "arm-07",
    "warehouseId": "ATL-01",
    "taskContext": "novel_package_scan",
    "taskOutcome": "scan_success",
    "dimensions_mm": { "x": 350, "y": 210, "z": 155 },
    "estimatedWeight_g": 850
  }
}
```

3.3. MVP Data Type 2: LogisticsEventData

The schema for non-robotic but robotic-adjacent data, likely sourced from a Warehouse Management System (WMS) via an API connector in the MyData Agent.

```
{
  "schemaVersion": "1.0",
  "eventTimestamp": "2025-07-12T19:52:10Z",
  "eventType": "INBOUND_SCAN",
  "locationId": "receiving-dock-03",
  "itemIdentifier": {
    "type": "UPC",
    "value": "012345678905"
  },
  "metadata": {
    "operatorId": "emp-451",
    "equipmentUsed": "handheld_scanner_zebra_mc9300",
    "discrepancyCode": null // e.g., 'DAMAGED_BOX', 'WRONG_ITEM'
  }
}
```

4. Agent Architecture & Interaction Flow

4.1. Agent Discovery & Lease Interaction Flow

The following sequence diagram illustrates the end-to-end flow for a data lease transaction.

```
sequenceDiagram
    participant Spender as Buyer-Side Agent
    participant DHT as KAD-DHT
    participant Earner as MyData Agent
    participant Polygon as Settlement Layer

    Spender->>DHT: Query for DataProducts with keyword "lidar"
    DHT-->>Spender: Return list of matching DataProducts & peer IDs
    Spender->>Earner: Initiate direct connection (libp2p)
    Spender->>Earner: Request lease for `productId`
    Earner->>Earner: Evaluate request against local policy (OPA)
    Note over Earner: Manual Approval: Send push notification to user
    Earner-->>Spender: Approve lease terms (price from PDVF)
    Spender->>Polygon: Call `createLease()` with terms & payment
    Polygon-->>Spender: LeaseCreated event with `leaseId`
    Spender->>Earner: Send `leaseId` as proof of payment
```

```
Earner->>Polygon: Verify `leaseId` on-chain
Note over Earner, Spender: Execute Federated Analysis of Features (PySyft)
Earner->>Spender: Send aggregated results
Spender->>Polygon: Finalize lease (optional, can be automated)
Polygon->>Earner: Release funds
Polygon->>Polygon: Record transaction for Royalty & Reputation calculation
```

4.2. MyData Agent - WMS Connector Architecture

The integration with Warehouse Management Systems will follow a pragmatic, two-phase approach:

- **Phase 1 (MVP): Direct API Adapter.** For our initial warehouse logistics partner, we will build a custom, point-to-point adapter. This adapter will be a module within the MyData Agent that connects directly to the partner's specific WMS API (e.g., REST, SOAP), authenticates, and transforms their proprietary event data into our canonical LogisticsEventData format. This minimizes initial complexity and accelerates time-to-market.
- **Phase 2 (Post-MVP): Middleware Pattern.** As we onboard more partners, we will refactor the connector architecture to use a middleware pattern. The MyData Agent will communicate with a central, protocol-level service that manages multiple adapters and exposes a single, unified API. This aligns with enterprise best practices and ensures long-term scalability and maintainability.

5. Smart Contract Interfaces (Polygon PoS - MVP)

This section defines the primary functions for the core smart contracts. These are simplified for clarity.

5.1. LeaseAgreement.sol

```
interface ILeaseAgreement {
    // Events
    event LeaseCreated(bytes32 leaseId, address spender, address earner, uint256 price);
    event LeaseApproved(bytes32 leaseId);
    event LeaseExecuted(bytes32 leaseId);

    // Functions
    function createLease(address earner, bytes32 dataProductId, uint256 maxPrice) external payable;
```

```
    function approveLease(bytes32 leaseId) external; // Called by Earner
    function executeLease(bytes32 leaseId) external; // Called by Spender
    function raiseDispute(bytes32 leaseId, string calldata reason) external;
}
```

5.2. RoyaltyDistributor.sol

```
interface IRoyaltyDistributor {
    // Events
    event RoyaltiesRecorded(bytes32 dataProductId, uint256 amount);
    event RoyaltiesClaimed(address earner, uint256 amount);

    // Functions
    function recordRoyalties(bytes32 leaseId, bytes32 dataProductId) external payable;
    function claimRoyalties() external; // Batched claim for a user
}
```

5.3. Reputation.sol

```
interface IReputation {
    // Functions
    function updateReputation(address user, bool successfulLease) external;
    function getReputation(address user) external view returns (uint256);
}
```

6. Security Considerations (Initial Threat Model)

- **Data Poisoning:**
  - **Threat:** An Earner provides malicious or garbage data to sabotage a Spender's AI model.
  - **Mitigation (MVP):** The Reputation contract will penalize bad actors. The sampleCid in the DataProduct allows for pre-lease verification. The Pandacea Arbitration Court (PAC) provides a mechanism for recourse.
- **Sybil Attacks on DHT:**
  - **Threat:** An attacker creates thousands of fake nodes to pollute the DHT or eclipse a specific user.
  - **Mitigation (MVP):** Libp2p has some built-in resistance. We will require a small PGT stake to publish a DataProduct, making Sybil attacks economically

infeasible.
- **Smart Contract Exploits:**
  - **Threat:** Reentrancy attacks, oracle manipulation, etc.
  - **Mitigation (MVP):** Adherence to Checks-Effects-Interactions pattern, use of OpenZeppelin audited base contracts, and a mandatory comprehensive third-party audit before mainnet launch.

7. Resolved Questions & Decisions (v1.1)

This section documents the resolution of the open questions from v1.0 based on the recent research findings.

- **Q1: Data Payload Size:** What are the expected file sizes for 3D sensor data?
  - **Finding:** File sizes can range from <1 MB for a single binary scan to over 1 GB for a long-duration, high-resolution scan.
  - **Decision:** The protocol must handle a wide range of file sizes. We will mandate the use of compressed binary formats (e.g., binary PCD, LAZ) to minimize storage. For payloads over a 10 MB threshold, the MyData Agent will be responsible for negotiating and funding a Filecoin storage deal to ensure persistent availability, with the deal ID being recorded on-chain as part of the lease.
- **Q2: WMS Connector Standardization:** How do we build a generic connector for multiple proprietary WMS APIs?
  - **Finding:** A middleware/iPaaS approach with a canonical data model is the scalable, long-term solution, but point-to-point adapters are faster for a single integration.
  - **Decision:** We will adopt a two-phase strategy as detailed in Section 4.2. The MVP will use a direct adapter for our first partner to ensure speed and focus, with the architecture designed for a future migration to a more scalable middleware pattern.
- **Q3: PySyft Performance:** What is the performance overhead for executing PySyft tasks on warehouse hardware?
  - **Finding:** PySyft can have significant overhead (~2x slower), and performance on constrained hardware (industrial PCs) is a major concern for complex models like CNNs. Network conditions are also a critical factor.
  - **Decision:** To de-risk the MVP, our initial implementation of the Privacy Layer will not focus on end-to-end federated training of large models from raw data. Instead, we will prioritize **federated analytics on pre-processed features**. For example, the MyData Agent can locally extract key features (e.g., object dimensions, centroids) from the 3D scans, and the federated task will be to

compute aggregate statistics or train simpler models on these much smaller feature vectors. This approach drastically reduces the compute, memory, and network burden while still providing significant privacy-preserving value.