PolicyManager Contract Comparison Report

This document summarizes the key differences between two Solidity smart contracts for managing insurance policies: `PolicyManager_v1.sol` (legacy version) and `PolicyManager_v2.sol` (updated version).

✅Overview

Feature	Version 2 (v2)	Version 1 (v1)	
	Author	Youssef	Mostafa
Policy Type	Crop insurance with seasonal handling	Generic insurance	
Subscription Deadline	✔️Yes	❌No	
Seasonal Support	✔️Yes (season, start/end, renewal, full vs sub-season)	❌No	
Treasury Contract	✔️Uses external ITreasury interface	❌Not used	
Payout Mechanism	Triggered by external engine, only after marking	Direct payout by owner	
Engine Integration	✔️External payout engine contract	❌Not present	
Subscribers Record	`currentSubscribers[]` with mapping for last season	`subscribers[]` only	
Full-Season Exclusivity	✔️Ensures single full-season policy per farmer	❌Not handled	
Upgradeable Seasons	✔️Resets state for new season	❌Not supported	
Historical Tracking	✔️Keeps `historicalSubscribers` per season	❌Not available	
Subscription Metadata	✔️Includes timestamped `Subscription` struct	❌Flat address array only	
Events	Detailed (Created, Subscribed, Payout, StatusChanged, SeasonReset)	Basic (Created, Subscribed, Payout, StatusChanged)	
Validation Modifiers	`onlyOwner`, `validPolicy`, `onlyPayoutEngine`	`onlyOwner`, `validPolicy`	

🛄Advanced Features in `v2`

™Seasonal Policy Management

- Includes: `season`, `seasonStart`, `seasonEnd`, and `subscriptionDeadline`
- Supports resubscription across seasons

🔄Full vs Partial Coverage Logic

- `coversFullSeason` boolean to control exclusivity
- Mapping to prevent sub-policy if full-season is already selected

Subscription Tracking

- Each farmer's subscription is stored as a `Subscription` struct with:
- `policyId`
- `timestamp`
- Subscriptions are grouped per `season`

🈺Historical Subscribers

- Uses `historicalSubscribers[policyId][season]` to archive each season's subscribers

🆒Integration with External Treasury

- Uses:

```
interface ITreasury {
    function deposit(address farmer) external payable;
}
```

- Sends premiums to a treasury instead of holding contract balance

Controlled Payout Flow

- `payoutEngine` contract must explicitly call `markPolicyAsPayout()`
- Ensures external conditions (e.g. oracle feeds) can trigger payout securely

🔽Simplified Features in `v1`

- Fixed payout based on `address(this).balance`
- No external calls, oracles, or seasonal dynamics
- Meant for educational/demonstration use

Let me know if you'd like to auto-generate a full `README.md` for your repo or setup GitHub Actions for deployment testing.