EB-SCC-005 · DEVELOPER SERIES

Smart Contract Cookbook

Production recipes for building on the Earth Blockchain Standards: fungible and non-fungible assets, regulated and identity-gated instruments, verifiable credentials and soulbound records, token-bound accounts for passports and AI agents, account-abstraction patterns, security patterns, and the audit-readiness checklist.

SOLIDITY · EBS RECIPES · ACCOUNT ABSTRACTION · SECURITY PATTERNS · AUDIT CHECKLIST

Version 1.0 · **Status** Public Release · **Date** July 2026

Canonical source Earth Blockchain Whitepaper v2.0

Audience Solidity engineers · protocol integrators · security reviewers

EB-SCC-005 · DEVELOPER SERIES · VERSION 1.0 · JULY 2026

Smart Contract Cookbook

PRODUCTION RECIPES FOR THE EARTH BLOCKCHAIN STANDARDS

How to Use This Cookbook

Each recipe answers one question — *how do I build X correctly on Earth Blockchain?* — with the standard to use, the pattern, working Solidity, and the pitfalls. Recipes assume the environment of EB-DEV-002 (toolchain, networks, verification) and compose the audited registry implementations of the nine EBS standards (Whitepaper §8; EB-SPEC-001 §8.5). Code targets Solidity 0.8.24 and is illustrative: adapt, test, and audit before production.

Prime directive. Deploy through `EBSRegistry.latest(standard)` clones, extend via hooks and modules, and never fork a standard's core — forks forfeit custodian, wallet, product, and auditor interoperability across the ecosystem (Whitepaper §8; EB-DEV-002 §7).

Table of Contents

1. Foundations: Registry, Clones, Roles, Upgrades

THE PATTERN EVERY RECIPE SHARES

```
import {Clones} from "@openzeppelin/contracts/proxy/Clones.sol";

IEBSRegistry constant REG = IEBSRegistry(EBS_REGISTRY);   // predeploy, resolve at deploy time

function deployAsset(string memory std, bytes memory initCall) returns (address a) {
    a = Clones.clone(REG.latest(std));                    // audited implementation, EIP-1167
    (bool ok,) = a.call(initCall);  require(ok, "init");
}
```

- **Roles.** Reference implementations use role-based access (admin, minter, pauser, controller…). Grant roles to a multisig/MPC, never an EOA; emit and document every grant.
- **Upgrades.** Clones are immutable by design — upgrade by deploying against a newer registry version and migrating, or wrap in a UUPS proxy *only* when governance genuinely needs mutability, with the proxy admin behind a timelock (mirror the protocol's own 48 h discipline, EB-SPEC-001 §9).
- **Finality in contracts.** Contracts never need reorg defenses here — but off-chain actors reacting to your events should follow EB-DEV-002 §5.

2. Fungible Assets (EBS-20) and wEARTH

RECIPE: SETTLEMENT UNIT WITH POLICY HOOKS

Use when: loyalty/settlement units, pooled environmental units, wrapped bridge assets (Whitepaper §8.1). **Remember:** EARTH itself is protocol-native — use the wEARTH predeploy for contract composability, never a bespoke wrapper.

```
contract SettlementUnit is EBS20Base {                  // registry implementation base
    function initialize(string n, string s, address admin) external initializer {
        __EBS20_init(n, s); _grantRole(DEFAULT_ADMIN_ROLE, admin);
    }
    // policy hook: allow-list enforcement without forking transfer logic
    function _beforeTokenTransfer(address from, address to, uint256 amt)
        internal override onlyAllowed(from, to) {}
}
```

Pitfalls. Don’t add fee-on-transfer semantics (breaks integrators’ accounting); don’t mint from constructors (clones have none — use `initialize` with `initializer` guard); pair pausability with a written unpause policy.

3. Unique, Batched, and Semi-Fungible (EBS-721 / 1155 / 3525)

RECIPES: SERIALIZED ITEMS, VINTAGED CLASSES, TRANCHE INSTRUMENTS

3.1 Serialized items (EBS-721)

Use when: one identity per physical or logical item — components, titles, one-of-one certificates (EarthSCM/Earth RWA patterns, Whitepaper §12). Commit content integrity on-chain, payload off-chain:

```
function mintItem(address to, uint256 id, bytes32 contentHash, string uri) external
onlyRole(MINTER) {
    _safeMint(to, id);
    _setContentCommitment(id, contentHash);   // verify off-chain metadata against this
    _setTokenURI(id, uri);
}
```

3.2 Batched classes (EBS-1155)

Use when: many units per class, classes keyed by meaning — EarthESG mints carbon credits as `classId = H(projectId || vintage)` (Whitepaper §12.2). Batch mint/transfer for gas efficiency; retirement is a burn that emits a certificate event consumed by reporting.

3.3 Tranches and divisible quantity (EBS-3525)

Use when: instruments unique in identity but fungible in quantity within a slot — bonds by tranche, invoices by debtor/maturity, credit lots that split and merge (Whitepaper §8.1). Core operations: `transferValue(fromTokenId, toTokenId, value)` within one slot; enforce slot-compatibility checks in hooks, never by trusting callers.

4. Regulated Instruments (EBS-1400 / 3643)

RECIPE: COMPLIANCE AS DETERMINISTIC CONTRACT LOGIC

Use when: securities-like instruments (EBS-1400: partitions, document binding, controller ops) or identity-gated assets (EBS-3643: transfers validated against an on-chain claims registry) — the Earth RWA rails (Whitepaper §8.1, §12.4).

```
// EBS-3643 pattern: every transfer consults the identity registry
function _beforeTokenTransfer(address from, address to, uint256 amt) internal override {
    require(identityRegistry.isEligible(to,   assetPolicyId), "TO_NOT_ELIGIBLE");
    require(identityRegistry.isEligible(from, assetPolicyId), "FROM_NOT_ELIGIBLE");
}
// EBS-1400 pattern: bind the governing document at issuance
setDocument("prospectus", "ipfs://…", 0xDOC_HASH);      // hash is the legal anchor
```

- **Claims, not addresses.** Gate on registry claims (jurisdiction, accreditation, KYC status issued by approved providers), never on hardcoded allow-lists — claims are revocable and auditable.
- **Controller operations.** Forced transfer/redemption exists for court-ordered actions; assign the controller role to the custodian/trustee under a documented mandate, and emit the mandated reason code on every use.
- **Error signaling.** Return the standard restriction codes so wallets and venues can explain *why* a transfer failed.

5. Credentials and Soulbound Records (EBS-4671 / 5192)

RECIPE: ATTESTATIONS THAT VERIFY WITHOUT THE ISSUER

Use when: degrees, licenses, verifier sign-offs, audit findings (EBS-4671, revocable, non-tradable) and permanently bound records — identity badges, compliance flags, agent reputation (EBS-5192) (Whitepaper §8.1, §12.1).

```
contract Diploma is EBS4671Base {
    function issue(address holder, bytes32 credHash, uint64 validUntil)
        external onlyRole(ISSUER) returns (uint256 id)
    { id = _mint(holder, credHash, validUntil); }        // non-transferable by base

    function revoke(uint256 id, bytes32 reason) external onlyRole(ISSUER) { _revoke(id, reason); }
}
```

- **Hash the credential, not the person.** On-chain: commitment + validity + revocation. Off-chain: the document, in issuer custody (EarthDoc pattern; selective disclosure per Whitepaper §12.1).
- **5192 for structure.** If transferability must be impossible by construction (not by policy), use EBS-5192 — e.g., sanctions flags consumed by §4 registries, and EarthAI reputation records.
- **Enumerate for verifiers.** Keep holder-indexed enumeration efficient; verifiers should resolve “does holder H hold a valid credential of type T from issuer I” in one call.

6. Token-Bound Accounts (EBS-6551)

RECIPE: ASSETS THAT ACT — PASSPORTS AND AGENTS

Use when: an EBS-721 identity must own things and execute — digital product passports holding their inspection credentials (EarthSCM), and AI agent roots holding wallet, mandates, and reputation (EarthAI) (Whitepaper §8.1, §12.3, §12.5).

```
address acct = registry6551.account(implementation, chainId, nftContract, tokenId, salt);
registry6551.createAccount(implementation, chainId, nftContract, tokenId, salt);

// the account executes only what the current NFT holder (or its policy) authorizes
IERC6551Account(acct).execute(target, value, data, 0);
```

- **Control follows the NFT.** Transferring the identity transfers control of everything the account holds — by design. For agents, the principal holds the NFT; for passports, the current custodian does.
- **Policy at the account.** Wrap `execute` with policy checks (spend limits, counterparty classes, mandate expiry from EBS-4671 credentials) — the enforcement point of the EarthAI trust stack (§7.2).

7. Account-Abstraction Patterns

SPONSORSHIP · SESSION KEYS · AGENT MANDATES

7.1 Sponsor paymaster

Your dApp pays gas so users transact in product terms (EB-DEV-002 §6.3). Validate *what* you sponsor: bind the paymaster to your contract selectors and per-account daily budgets; anything looser becomes a gas faucet.

7.2 Agent session pattern (EarthAI substrate)

```
// smart-account validation hook: enforce a live mandate before any agent action
function _validate(UserOp calldata op) internal view {
    Mandate memory m = mandates.currentOf(agentRoot);        // EBS-4671 credential
    require(block.timestamp < m.expiry,                "MANDATE_EXPIRED");
    require(spentToday + op.value <= m.dailyLimitGr,   "OVER_LIMIT");
    require(m.allowedTargets[op.target],               "TARGET_FORBIDDEN");
}
```

Revoking the credential kills authority instantly while history remains attributable — the property Whitepaper §12.5 promises. Session keys for human users are the same pattern with a device key instead of an agent root.

8. Security Patterns and Anti-Patterns

THE SHORT LIST THAT PREVENTS LONG INCIDENTS

Do	Don’t
Checks-effects-interactions; reentrancy guards on value-moving externals.	Call untrusted contracts before state is settled.
Pull payments for distributions (claim pattern, as the protocol itself does — EB-SPEC-001 §7.3).	Push loops over unbounded holder sets.
Resolve system/standard addresses from EBSRegistry at deploy time.	Hardcode addresses or assume cross-network constancy.
Role admin behind multisig + timelock; events on every privileged action.	EOA admins; silent owner powers.
Custom errors + restriction codes (§4) for failed transfers.	Bare `require(false)` that venues can’t interpret.
Invariant tests: supply conservation, role reachability, pause recoverability.	Only happy-path unit tests.
Treat oracle and bridge inputs as adversarial; verify certificates where offered (EB-API-003 §7).	Trust “finalized” claims from unverified endpoints.

9. Testing and Audit-Readiness Checklist

BEFORE MAINNET

- **Coverage.** Unit + property tests (Foundry invariants) on the devnet image (EB-DEV-002 §9); fork tests against testnet registry state.
- **Static analysis.** Slither/solhint clean or triaged; compiler warnings zero; pinned toolchain committed.
- **Standards conformance.** Interface IDs and behavior verified against the registry implementation's conformance suite for each EBS used.
- **Access review.** A written table of every role, holder, and rotation procedure; deployment scripts idempotent and reviewed.
- **Verification & docs.** Source verified on scan.earth.ooo; NatSpec on external functions; the standard-JSON build input archived (EB-DEV-002 §4.3).
- **External audit.** Independent review for anything holding user assets, with findings and resolutions published — the ecosystem norm the enterprise products themselves follow.

References

SUITE CROSS-REFERENCES

[1] Earth Blockchain Whitepaper v2.0 — EBS standards (§8), product patterns (§12).

[2] EB-SPEC-001 — EBSRegistry and system contracts (§8), governance timelock discipline (§9).

[3] EB-DEV-002 — environment, verification, finality patterns, sponsorship overview.

[4] EB-API-003 — issuance API, webhook verification, light-client proofs.

[5] OpenZeppelin Contracts and the ERC corpus (eips.ethereum.org) — informative baselines for interface lineage.