EB-DEV-002 · DEVELOPER SERIES

# Developer Documentation

The practical guide to building on Earth Blockchain: connecting to the network, accounts and wallets, deploying and verifying smart contracts with standard EVM tooling, finality-aware application patterns, gas and fee handling, EBS standards integration, events and indexing through BaaS, and testing from local devnet to mainnet.

EVM TOOLING · JSON-RPC · HARDHAT · FOUNDRY · EBS STANDARDS · BAAS APIS

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

**Canonical source** Earth Blockchain Whitepaper v2.0

**Audience** dApp developers · integration engineers · Web3 teams

EB-DEV-002 · DEVELOPER SERIES · VERSION 1.0 · JULY 2026

# Developer Documentation

BUILDING ON EARTH BLOCKCHAIN WITH STANDARD EVM TOOLING

## About This Guide

This guide takes a developer from zero to a production integration on Earth Blockchain. Because the network is fully EVM-compatible (Whitepaper v2.0 §7; EB-SPEC-001 §6), everything you already know — Solidity, Hardhat, Foundry, ethers.js, viem, MetaMask-class wallets — works unchanged. What this guide adds is the Earth-specific knowledge that makes applications *better* here: how to exploit deterministic finality instead of counting confirmations, how the base-fee market behaves at 3-second slots, how to consume the `earth_` RPC namespace, how to build on EBS standards rather than bespoke contracts, and how to integrate through BaaS when you would rather not run infrastructure.

This document does not restate protocol design (see the Whitepaper and EB-SPEC-001), does not provide full API schemas (EB-API-003), and does not cover validator operations (EB-VAL-004) or deep contract recipes (EB-SCC-005).

**Endpoint and chain-ID convention.** Examples use the canonical endpoint names `https://rpc.earth.ooo` (mainnet), `https://rpc.testnet.earth.ooo` (testnet), and the placeholder `<CHAIN_ID>`. Live values, the EIP-155 chain identifiers, and genesis artifacts are published at earth.ooo and on the explorer, scan.earth.ooo. Always confirm against the published genesis artifacts before signing.

## Table of Contents

1. Networks and Connectivity ..... 2
2. Accounts, Wallets, and Keys ..... 3
3. Quickstart: First Transaction in Five Minutes ..... 4
4. Smart Contract Development and Verification ..... 5
5. Finality-Aware Application Design ..... 6
6. Gas, Fees, and Sponsorship ..... 7
7. Building with EBS Standards ..... 8
8. Events, Indexing, and BaaS Integration ..... 9
9. Testing: Devnet, Testnet, CI ..... 10
10. Troubleshooting Reference ..... 11
11. Best Practices and Security Checklist ..... 12
References ..... 13

EARTH BLOCKCHAIN

# 1. Networks and Connectivity

MAINNET · TESTNET · LOCAL DEVNET · EXPLORER

| Network | RPC (canonical name) | Purpose |
|---|---|---|
| **Mainnet** | https://rpc.earth.ooo | Production. EARTH has real value; deployments are permanent. |
| **Testnet** | https://rpc.testnet.earth.ooo | Faucet-funded staging with mainnet-identical parameters (EB-SPEC-001 §13). |
| **Local devnet** | http://127.0.0.1:8545 | Single-node PoSA image with instant finality for unit and CI testing (§9). |

Three ways to connect, in increasing order of operational ownership:

1. **BaaS managed endpoints** (recommended for enterprises and most dApps): SLA-backed JSON-RPC, WebSocket streams, and webhook delivery without running nodes (Whitepaper §7.5). Authentication and quotas per EB-API-003.
2. **Public community endpoints:** best-effort RPC for development and light traffic.
3. **Self-hosted full node:** full sovereignty; hardware guidance in EB-VAL-004 §2 applies with consensus duties omitted.

The explorer at **scan.earth.ooo** provides contract verification (§4.3), the validator set, quorum-certificate inspection, and epoch economics — the visual counterpart of the `earth_` RPC namespace.

**Fig. 1 — Developer Path: Local → Testnet → Mainnet**


*Identical protocol parameters at every stage — behavior observed on testnet is behavior shipped on mainnet.*

*Fig. 1. The promotion path. Because parameters match EB-SPEC-001 §13 across environments, no stage-specific code paths are needed.*

EARTH BLOCKCHAIN

# 2. Accounts, Wallets, and Keys

EOAS · EARTH WALLETS · ACCOUNT ABSTRACTION

Accounts are standard EVM accounts: secp256k1 EOAs and contract accounts, EIP-55 checksummed addresses, EIP-155 replay-protected signatures (EB-SPEC-001 §2). Any EVM wallet can hold EARTH and sign for Earth Blockchain once the network (RPC URL + chain ID) is added.

| Option | When to Use |
|---|---|
| **Standard EVM wallets** | Individual developers and users; add the network manually or via the wallet-config link on earth.ooo. |
| **Earth Wallets — non-custodial** | First-party browser/mobile wallets with native display of finality status, EBS assets, and staking positions (Whitepaper §7.4). |
| **Earth Wallets — custodial / MPC** | Enterprise key management: role-based controls, transaction policies, approval workflows. Covered operationally in EB-ENT-006. |
| **Account abstraction** | Smart accounts with session keys, spending limits, and gas sponsorship — the substrate for consumer UX and EarthAI agent accounts (Whitepaper §7.4, §12.5). Patterns in §6.3 and EB-SCC-005. |

**Key hygiene.** Never place private keys in source or CI variables in plaintext; use environment-injected signers or remote signing. For anything holding meaningful EARTH, prefer hardware or MPC custody. Deployment keys SHOULD be distinct from treasury keys and rotated after mainnet launch.

EARTH BLOCKCHAIN

# 3. Quickstart: First Transaction in Five Minutes

CONNECT · FUND · SEND · CONFIRM FINALITY

**Step 1 — connect.** Using viem (TypeScript); ethers.js is equivalent:

```
import { createPublicClient, createWalletClient, http, parseEther } from 'viem'
import { privateKeyToAccount } from 'viem/accounts'

const earth = {                                  // chain object — values from genesis artifacts
  id: Number(process.env.CHAIN_ID),
  name: 'Earth Blockchain',
  nativeCurrency: { name: 'EARTH', symbol: 'EARTH', decimals: 18 },
  rpcUrls: { default: { http: ['https://rpc.testnet.earth.ooo'] } },
}
const pub    = createPublicClient({ chain: earth, transport: http() })
const signer = createWalletClient({ chain: earth, transport: http(),
                 account: privateKeyToAccount(process.env.PK as `0x${'{'}string{'}'}`) })
```

**Step 2 — fund.** Request testnet EARTH from the faucet linked at earth.ooo (rate-limited per address; CI faucet keys available through the developer program, Whitepaper §7.6).

**Step 3 — send and confirm.**

```
const hash = await signer.sendTransaction({ to: '0xRecipient…', value: parseEther('1') })
const rcpt = await pub.waitForTransactionReceipt({ hash })   // included ≈ next 3 s slot

// Earth-specific: confirm deterministic finality (one RPC call, no confirmation counting)
const finalized = await pub.request({ method: 'earth_getFinalizedHead' }) as { number:
`0x${'{'}string{'}'}` }
const isFinal   = BigInt(finalized.number) >= rcpt.blockNumber
console.log({ included: rcpt.status === 'success', isFinal })
```

On Earth Blockchain the gap between inclusion and finality is one quorum certificate — typically the next slot (Whitepaper §6.4; EB-SPEC-001 §5.5). Section 5 turns this into application patterns.

# 4. Smart Contract Development and Verification

HARDHAT · FOUNDRY · EXPLORER VERIFICATION

## 4.1 Hardhat

```
// hardhat.config.ts
export default {
  solidity: { version: '0.8.24', settings: { optimizer: { enabled: true, runs: 200 } } },
  networks: {
    earthTestnet: { url: 'https://rpc.testnet.earth.ooo',
                    chainId: Number(process.env.CHAIN_ID_TESTNET),
                    accounts: [process.env.DEPLOYER_PK!] },
    earth:        { url: 'https://rpc.earth.ooo',
                    chainId: Number(process.env.CHAIN_ID),
                    accounts: [process.env.DEPLOYER_PK!] },
  },
}
```

`npx hardhat run scripts/deploy.ts --network earthTestnet` deploys exactly as on any EVM chain. Because blocks are 3 s and finality deterministic, deployment scripts SHOULD gate follow-on transactions on `earth_getFinalizedHead` rather than fixed sleeps.

## 4.2 Foundry

```
# foundry.toml — rpc endpoints
[rpc_endpoints]
earth_testnet = "https://rpc.testnet.earth.ooo"
earth         = "https://rpc.earth.ooo"

# deploy + broadcast
forge create src/MyContract.sol:MyContract \
  --rpc-url earth_testnet --private-key $DEPLOYER_PK
```

## 4.3 Verification on scan.earth.ooo

The explorer accepts standard-JSON verification (single command from both toolchains). Verified source is a de-facto requirement for enterprise counterparties and for EBS-registry listing (§7):

```
npx hardhat verify --network earthTestnet 0xDeployedAddress "constructorArg1"
forge verify-contract 0xDeployedAddress src/MyContract.sol:MyContract \
  --verifier-url https://scan.earth.ooo/api --chain-id $CHAIN_ID_TESTNET
```

**Reproducibility.** Pin compiler versions and optimizer settings in the repo; verification requires byte-exact metadata. Commit the standard-JSON input used for mainnet deployments — auditors and the EBS registry review it.

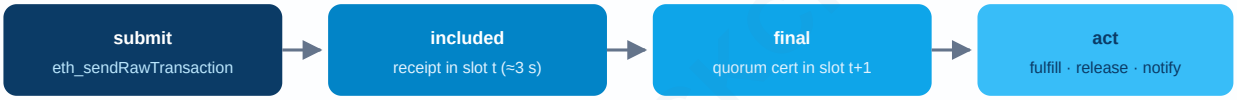
# 5. Finality-Aware Application Design

DESIGNING FOR DETERMINISTIC SETTLEMENT

Most EVM application code is written defensively against reorgs: N-confirmation waits, reorg handlers in indexers, "pending vs. safe" UI states. On Earth Blockchain a block covered by a quorum certificate is *final* — irreversible under the protocol's fault bound (EB-SPEC-001 §5.5) — so that machinery collapses into one check.

| Legacy Pattern | Earth Blockchain Pattern |
|---|---|
| **Wait N confirmations before acting** | Act when `block ≤ earth_getFinalizedHead()` — usually the next slot. |
| **Reorg handlers in indexers** | Index only finalized blocks; no rollback path is required (§8). |
| **"Pending / probably settled" UX** | Binary UX: *submitted* → *final*. Show finality, not confidence. |
| **Off-chain settlement delay for payments** | Treat the finality receipt as settlement for order fulfillment and machine payments (Whitepaper §6.4, §12.5). |

**Fig. 2 — Application-Side Transaction Lifecycle**




*End-to-end ≈ 3–6 s. The only state your application must model after inclusion is "final".*

*Fig. 2. The two-state lifecycle applications should implement, replacing confirmation counting.*

For maximum assurance — e.g., a bridge, exchange, or auditor — verify the certificate itself: fetch `earth_getQuorumCert(blockHash)` and check ≥ ⌈2n/3⌉ signers against `earth_getValidatorSet(epoch)`. Light-client libraries in the SDK wrap this (EB-API-003 §5).

EARTH BLOCKCHAIN

# 6. Gas, Fees, and Sponsorship

TYPE-2 TRANSACTIONS · ESTIMATION · GAS ABSTRACTION

## 6.1 Fee Mechanics for Developers

Fees follow the base-fee + tip model (EB-SPEC-001 §6.2): you pay `gasUsed × (baseFee + tip)`, and a governed share of the base fee is burned (β = 0.30 at genesis, Whitepaper §15.6). Practical consequences:

- **Fees are damped by design.** The base fee moves ≤ 12.5% per 3-second block; short congestion spikes decay in seconds, so aggressive fee-bumping loops are rarely needed.
- **Set maxFee with headroom, tip modestly.** A common policy: `maxFeePerGas = 2 × current baseFee + tip`; unspent headroom is refunded. Default tips are sufficient except when competing within a single slot.
- **Estimate normally.** `eth_estimateGas` and `eth_feeHistory` behave as on any EVM chain.

## 6.2 Enterprise Settlement via BaaS

Applications integrating through BaaS may settle metered usage in EARTH at contracted rates instead of handling per-transaction fees (Whitepaper §7.2) — the recommended posture for line-of-business systems whose owners should never see a gas dialog. Configuration is per-project in the BaaS console (EB-API-003 §2).

## 6.3 Gas Sponsorship and Session Keys

Account-abstraction support enables sponsored transactions (your dApp pays), session keys (bounded authority for a device or agent), and spending limits — the same primitives EarthAI mandates build on (Whitepaper §12.5). Recipe-level implementations, including a sponsor-paymaster pattern and an agent session-key pattern, are in EB-SCC-005 §7.

EARTH BLOCKCHAIN

# 7. Building with EBS Standards

### COMPOSE THE REGISTRY, DON'T REINVENT CONTRACTS

The nine Earth Blockchain Standards (Whitepaper §8) are protocol-registered, audited reference implementations, interface-compatible with their ERC lineage — meaning OpenZeppelin-style tooling, wallets, and auditors already understand them. The EBSRegistry predeploy (EB-SPEC-001 §8.5) is the canonical source of implementation addresses and versions.

| Standard | Use It For | Deep-Dive Recipe |
|---|---|---|
| **EBS-20** | Fungible assets; wEARTH interop | EB-SCC-005 §2 |
| **EBS-721 / 1155 / 3525** | Unique, batched, and semi-fungible assets | EB-SCC-005 §3 |
| **EBS-1400 / 3643** | Regulated instruments; identity-gated transfers | EB-SCC-005 §4 |
| **EBS-4671 / 5192** | Credentials; soulbound records | EB-SCC-005 §5 |
| **EBS-6551** | Token-bound accounts (passports, agents) | EB-SCC-005 §6 |

Minimal issuance example against the registry (issuing an EBS-721 collection through the audited implementation rather than a hand-rolled contract):

```
IEBSRegistry reg = IEBSRegistry(EBS_REGISTRY);          // predeploy, EB-SPEC-001 §8.5
address impl = reg.latest("EBS-721");                    // audited reference implementation
MyCollection c = MyCollection(Clones.clone(impl));       // EIP-1167 minimal proxy
c.initialize("Serialized Components", "COMP", admin);    // role setup per EB-SCC-005 §3
```

**Why this matters commercially.** Custodians, auditors, and the enterprise products integrate against registry implementations. A bespoke fork forfeits that interoperability; extend via hooks and modules instead (EB-SCC-005 §1.3).

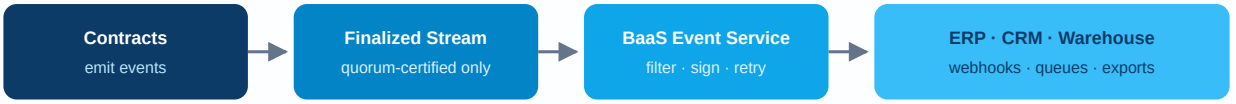
EARTH BLOCKCHAIN

# 8. Events, Indexing, and BaaS Integration

LOGS · STREAMS · WEBHOOKS · ANCHORING

Contract events are standard EVM logs, queryable via `eth_getLogs` and streamed over WebSocket. Earth Blockchain's difference is operational: index the **finalized** stream and your pipeline needs no reorg handling — every delivered event is settled truth (§5).

**Fig. 3 — Finalized Event Pipeline into Enterprise Systems**




*Webhook payloads carry block number, quorum-certificate reference, and an HMAC signature — verify both before acting.*

*Fig. 3. The reorg-free event pipeline: only finalized events leave the BaaS boundary (Whitepaper §7.5).*

**Anchoring.** For document- and data-integrity use cases, the BaaS anchoring API commits a hash on-chain in one call — the primitive beneath EarthDoc, EarthESG, and EarthSCM (Whitepaper §7.5). Applications SHOULD anchor content hashes, never content: data stays in your custody, verifiability becomes public.

**Self-hosted indexing.** Point standard EVM indexers at a full node and follow `earth_getFinalizedHead` as the cursor. Historical backfill uses an archive node (§1); epoch-boundary system transactions (rewards, set rotation) appear as regular transactions from the system address and can be filtered by that sender.

EARTH BLOCKCHAIN

# 9. Testing: Devnet, Testnet, CI

DETERMINISTIC LOCAL CHAINS · FAUCETS · PIPELINES

- **Unit tests.** Hardhat Network / Anvil remain ideal for pure-EVM logic. For finality-dependent logic (§5), use the **local devnet image** — a single-validator PoSA node with genesis parameters matching §13 of EB-SPEC-001 and per-block certificates, so `earth_*` calls behave as in production.
- **Fixtures.** The devnet ships with funded accounts, the system predeploys, and EBSRegistry populated — tests can clone reference implementations exactly as in §7.
- **CI.** Run the devnet as a service container; gate merges on a testnet smoke deployment. Faucet automation keys for CI are issued through the developer program (Whitepaper §7.6).
- **Rehearse upgrades.** Governance changes reach mainnet only through a 48 h timelock (EB-SPEC-001 §9); subscribe to the Timelock queue on testnet and mainnet and include a “parameters changed” scenario in integration tests.

EARTH BLOCKCHAIN

# 10. Troubleshooting Reference

SYMPTOMS → CAUSES → RESOLUTIONS

| Symptom | Likely Cause | Resolution |
|---|---|---|
| **“invalid chain id” on signing** | Wallet configured with wrong EIP-155 ID | Re-add network from genesis artifacts at earth.ooo; never hardcode guessed IDs. |
| **Tx stuck as pending** | maxFeePerGas below current base fee, or nonce gap | Re-send same nonce with fee headroom (§6.1); inspect account nonce via `eth_getTransactionCount(…, 'pending')`. |
| **Receipt present but app logic misfires** | Acting on inclusion instead of finality | Gate on `earth_getFinalizedHead` (§5); with BaaS webhooks this is automatic. |
| **Verification fails on scan.earth.ooo** | Compiler/optimizer metadata mismatch | Verify with the exact standard-JSON input used to build; pin versions (§4.3). |
| **Events missing in indexer** | Cursor follows head instead of finalized; or non-archive node for backfill | Use finalized cursor (§8); use archive node for history (§1). |
| **“replacement transaction underpriced”** | Bump below required increment | Increase both maxFee and tip ≥ 10% over the pending values. |
| **Contract call reverts only on mainnet** | Registry/predeploy address assumed rather than read | Resolve addresses through EBSRegistry and published genesis artifacts (§7; EB-SPEC-001 §8). |

EARTH BLOCKCHAIN

# 11. Best Practices and Security Checklist

### SHIP-READY CRITERIA

- **Finality discipline.** Every irreversible side effect (fulfillment, payout, credential issuance) gates on finality, not inclusion.
- **Address discipline.** System and standard addresses are resolved from EBSRegistry/genesis artifacts at startup — never compiled in.
- **Standards first.** Assets, credentials, and agent identities use EBS implementations; custom logic lives in extensions, keeping custodian and product interoperability (§7).
- **Key separation.** Deployer ≠ admin ≠ treasury; admin roles behind multisig or MPC; renounce or timelock where possible.
- **Upgrade awareness.** Monitor the EarthDAO Timelock queue; treat queued proposals as 48-hour change notices (EB-SPEC-001 §9).
- **Webhook verification.** Validate HMAC signatures and the referenced quorum certificate on BaaS deliveries before mutating state (§8).
- **Audit posture.** Verified source on scan.earth.ooo, pinned toolchain, reproducible builds, and third-party review for anything holding user assets.
- **Fee UX.** Consumer flows use sponsorship or BaaS settlement so end users transact in your product's terms, not in gas (§6).

EARTH BLOCKCHAIN

# References

## SUITE CROSS-REFERENCES AND EXTERNAL SOURCES

[1] Earth Blockchain Whitepaper v2.0 (July 2026) — canonical design reference. earth.ooo.

[2] EB-SPEC-001 — Technical Protocol Specification v1.0: consensus, finality, fees, system contracts, parameter registry.

[3] EB-API-003 — SDK & API Documentation: full JSON-RPC schemas, BaaS endpoints, client libraries.

[4] EB-SCC-005 — Smart Contract Cookbook: EBS recipes, account-abstraction patterns, security patterns.

[5] Earth Blockchain explorer — scan.earth.ooo: verification, validator set, quorum certificates, epoch economics.

[6] Ethereum tooling references informing examples: Hardhat, Foundry, viem, ethers.js documentation; EIP-155, EIP-1559 (eips.ethereum.org).

*EB-DEV-002 v1.0 (July 2026). Endpoint names and identifiers shown are canonical conventions; live values ship with the genesis artifacts at earth.ooo. Code samples are illustrative and unaudited; review before production use. This document does not constitute financial, investment, tax, or legal advice.* *© 2026 Earth Blockchain Corp. (Wyoming, USA) · Alpha Blockchain Private Limited (India). All rights reserved.*

EARTH BLOCKCHAIN