THE BITCOIN CORPORATION

Smart Contract Hierarchy

Blockchain Architecture & Contract Patterns

Document Version: 1.0 | Date: November 2025 | The Bitcoin Corporation LTD

Bitcoin Corporation Smart Contract Hierarchy

🏗️ Organizational Structure

```
github.com/bitcoin-corp/
├── bitcoin-os/ # Master OS Repository
│ ├── packages/
│ │ └── bitcoin-os-bridge/ # Shared components (npm package)
│ └── bitcoin-apps/ # Submodule linking to apps suite
│ ├── bitcoin-writer/ # Individual app repos
│ ├── bitcoin-email/
│ ├── bitcoin-music/
│ ├── bitcoin-wallet/
│ ├── bitcoin-drive/
│ └── ...
└── smart-contracts/ # Master contract templates
├── creator-contracts/
├── developer-contracts/
└── governance/
```

📜 Smart Contract Templates

1. Master Creator Contract (bitcoin-corp level)

```
solidity

contract MasterCreatorContract {

// Base template inherited by all apps

struct Creator {

address wallet;

string role; // "developer", "designer", "content", "tester"

uint256 contributionScore;

mapping(string => uint256) tokenBalances; // Multi-token holdings

}

mapping(address => Creator) public creators;

mapping(string => address) public tokenContracts; // App name -> Token contract

// Standard payout function inherited by all apps

function payoutTokens(

address creator,

string memory appName,

uint256 amount

) public onlyAuthorized {

IERC20(tokenContracts[appName]).transfer(creator, amount);

creators[creator].tokenBalances[appName] += amount;

}

}
```

2. Developer Contract Template

```
solidity

contract DeveloperContract is MasterCreatorContract {

struct Contribution {

string repoName; // "bitcoin-music", "bitcoin-writer", etc

string commitHash;

uint256 linesAdded;

uint256 linesRemoved;
uint256 complexity; // 1-10 scale

uint256 tokenReward;

bool approved;

}

mapping(address => Contribution[]) public contributions;

// Automatic token calculation

function calculateReward(

uint256 linesOfCode,

uint256 complexity,

string memory repoName

) public view returns (uint256) {

uint256 baseRate = getBaseRate(repoName);

return linesOfCode complexity baseRate;

}

// GitHub integration hook

function submitPR(

string memory repoName,

string memory commitHash,

uint256 linesAdded,

uint256 linesRemoved

) external {

// Automated from GitHub Actions

uint256 reward = calculateReward(

linesAdded,

estimateComplexity(repoName, commitHash),

repoName

);

// Issue tokens automatically

payoutTokens(msg.sender, repoName, reward);

}

}
```

3. Content Creator Contract

```
solidity

contract ContentCreatorContract is MasterCreatorContract {

struct Content {

string appName; // Which app they created content for

string contentType; // "music", "document", "video", etc

string ipfsHash; // Stored on IPFS

uint256 views;

uint256 revenue;

uint256 creatorShare; // Percentage

}

mapping(address => Content[]) public creatorContent;

// Revenue sharing

function distributeRevenue(

address creator,

string memory appName,

uint256 revenue

) external {

uint256 creatorPayout = (revenue * 70) / 100; // 70% to creator

uint256 appPayout = (revenue * 20) / 100; // 20% to app

uint256 corpPayout = (revenue * 10) / 100; // 10% to corp

// Pay in respective tokens

payoutTokens(creator, appName, creatorPayout);

payoutTokens(appTreasury[appName], appName, appPayout);

payoutTokens(corpTreasury, "bCorp", corpPayout);

}

}
```

🎯 Hierarchical Inheritance

Top-Down Structure

```
MasterCreatorContract (bitcoin-corp)

↓ inherits

DeveloperContract (bitcoin-os)

↓ inherits & customizes

AppDeveloperContract (bitcoin-music, bitcoin-writer, etc)
```

How Changes Propagate

1. **Update Master Contract** → All apps get new features

2. **Update OS Contract** → All apps under OS updated

3. **Update App Contract** → Only that app changes

🔄 Automatic Token Distribution

GitHub Actions Integration

```
yaml

.github/workflows/token-payout.yml
---

name: Automatic Token Payout

on:

pull_request:

types: [closed]

jobs:

payout:

if: github.event.pull_request.merged == true

runs-on: ubuntu-latest

steps:

- name: Calculate Contribution
```

```
run: |

LINES=$(git diff --numstat | awk '{added+=$1; removed+=$2} END {print added,
removed}')

REPO_NAME="${{ github.repository.name }}"

- name: Submit to Smart Contract

run: |

# Call smart contract with contribution data

contract.submitPR($REPO_NAME, $COMMIT_HASH, $LINES_ADDED, $LINES_REMOVED)
```

📊 Token Flow Hierarchy

```
Developer contributes to bitcoin-music

↓

GitHub Action triggers

↓

Smart Contract calculates reward

↓

Issues $bMusic tokens

↓

Records in MasterCreatorContract

↓

Can atomic swap to $bOS or $bCorp
```

💼 Standard Contract Types

1. Core Developer Contract

- **Tokens**: Based on code contribution
- **Metrics**: Lines, complexity, impact
- **Payout**: Immediate on PR merge

2. App Creator Contract

- **Tokens**: Based on app creation
- **Metrics**: User adoption, revenue
- **Payout**: Vesting schedule

3. Content Creator Contract

- **Tokens**: Revenue sharing
- **Metrics**: Views, engagement, sales
- **Payout**: Monthly based on performance

4. Bug Bounty Contract

- **Tokens**: Fixed amounts
- **Metrics**: Severity levels
- **Payout**: On verification

5. Documentation Contract

- **Tokens**: Per page/guide
- **Metrics**: Completeness, quality
- **Payout**: On review approval

🚀 Implementation Plan

Phase 1: Contract Deployment

```
bash

bitcoin-corp/

├── smart-contracts/

│ ├── MasterCreator.sol # Deploy first

│ ├── Developer.sol # Deploy second

│ └── templates/ # App-specific templates
```

Phase 2: App Integration

```
bash

bitcoin-os/

├── bitcoin-apps/

│ ├── bitcoin-music/

│ │ └── contracts/

│ │ └── MusicCreator.sol # Inherits from Developer.sol

│ ├── bitcoin-writer/

│ │ └── contracts/

│ │ └── WriterCreator.sol # Inherits from Developer.sol
```

Phase 3: Automation

- GitHub Actions for automatic payouts
- Contract verification systems
- Dashboard for tracking contributions

🔐 Governance Structure

Contract Update Process

1. **Proposal** → Submit change to MasterCreator
2. **Vote** → $bCorp holders vote
3. **Update** → Deploy new version
4. **Propagate** → All child contracts updated

Emergency Controls

- Pause mechanism for security
- Multi-sig for critical functions
- Upgrade proxy pattern

📈 Benefits of Hierarchical System

For Developers

- **Clear contracts** before contributing
- **Automatic payouts** on merge
- **Transparent calculations**
- **Multi-token earnings**

For Bitcoin Corp

- **Standardized agreements**
- **Automated administration**
- **Reduced overhead**
- **Scalable to 1000s of contributors**

For Apps

- **Inherit proven contracts**
- **Customize for specific needs**
- **Automatic GitHub integration**
- **Built-in revenue sharing**

🎯 Competitive Advantage

BSV/MetaNet: Manual payments, unclear terms

Bitcoin OS:

- Smart contracts = automatic payments
- Hierarchical = consistent standards
- Transparent = everyone knows the rules
- Immediate = merge PR, get tokens

📝 Example: Developer Journey

1. **John wants to contribute to Bitcoin Music**
2. Reviews `MusicCreator.sol` contract terms
3. Submits PR fixing audio player bug
4. PR merged → GitHub Action triggers
5. Smart contract calculates: 150 lines × 5 complexity = 750 $bMusic
6. Tokens sent to John's wallet immediately
7. John can hold $bMusic or swap to $bOS or $bCorp
8. John earns dividends from Bitcoin Music revenue

This creates the world's first automated, hierarchical, multi-token development economy!

Every contribution is contracted.

Every merge is compensated.

Every developer is an owner.

Every app inherits from the master.

No more volunteer coding. Welcome to professional open-source.