

Some useful implementation examples

Table of Contents

1. Core Implementation
2. Asset Journey Examples
3. Phase Transition Scenarios
4. Threshold Calculations
5. Governance Flows
6. Implementation Considerations

1. Core Implementation

Base Structure

```
data RealWorldAsset = RealWorldAsset {  
    assetId :: AssetId,  
    assetType :: AssetType,  
    assetValue :: Integer,  
    assetMetadata :: AssetMetadata,  
    custodialInfo :: CustodianInfo,  
    verificationData :: VerificationInfo,  
    requirementSet :: [Requirement]  
}  
  
-- Extensible asset types to accommodate any real-world asset  
data AssetType =  
    ArtPiece ArtDetails  
  | RealEstate PropertyDetails  
  | PreciousMetals MetalDetails  
  | Commodity CommodityDetails  
  | CustomAsset CustomDetails  
  
-- Requirement system that handles any type of requirement  
data Requirement = Requirement {  
    requirementType :: RequirementType,  
    description :: Text,
```

```

validationMethod :: ValidationMethod,
verificationFrequency :: Frequency,
requirementStatus :: RequirementStatus,
lastVerified :: Maybe POSIXTime
}

```

```

data RequirementType =

```

```

    Legal
  | Physical
  | Operational
  | Environmental
  | Financial
  | Custom Text

```

Agora Governance Integration

```

data AgoraPhase =

```

```

    LockPhase LockPhaseInfo
  | VotingPhase VotingPhaseInfo
  | ExecutionPhase ExecutionPhaseInfo

```

```

data GovernanceConfig = GovernanceConfig {

```

```

    thresholds :: AgoraThresholds,
    phaseDurations :: PhaseDurations,
    votingPowerStrategy :: VotingStrategy,
    effectValidators :: Map EffectType EffectValidator

```

```

}

```

```

data AgoraThresholds = AgoraThresholds {

```

```

    creation :: Percentage,  -- e.g., 5%
    start :: Percentage,    -- e.g., 10%
    vote :: Percentage,     -- e.g., 15%
    execution :: Percentage, -- e.g., 51%
    cosigning :: Percentage  -- e.g., 25%

```

```

}

```

```

data PhaseDurations = PhaseDurations {

```

```

    lockPhaseDuration :: POSIXTime,
    votingPhaseDuration :: POSIXTime,
    executionPhaseDuration :: POSIXTime

```

```
}
```

2. Asset Journey Examples

Example 1: Fine Art Tokenization

```
-- Picasso painting tokenization example
artExample :: RealWorldAsset
artExample = RealWorldAsset {
  assetId = "art_001",
  assetType = ArtPiece {
    artist = "Pablo Picasso",
    title = "Example Painting",
    year = 1937,
    medium = "Oil on canvas",
    dimensions = Dimensions 349.3 776.6,
    authenticity = [Certificate1, Certificate2]
  },
  assetValue = 10_000_000_000_000, -- 10M ADA
  requirementSet = [
    Requirement {
      requirementType = Physical,
      description = "Temperature control 20-22°C",
      validationMethod = TemperatureSensorValidation,
      verificationFrequency = Hourly,
      requirementStatus = Active
    },
    Requirement {
      requirementType = Legal,
      description = "Insurance coverage",
      validationMethod = InsuranceDocValidation,
      verificationFrequency = Monthly,
      requirementStatus = Active
    }
  ]
}

-- Art-specific proposal example
proposeMoveToNewGallery :: ProposalParams
proposeMoveToNewGallery = ProposalParams {
```

```

description = "Move artwork to Modern Gallery",
effect = PhysicalLocationChange {
    newLocation = "Modern Gallery, NY",
    transportMethod = "Specialized Art Transport",
    insuranceCoverage = "Extended Transit Insurance"
},
requiredCosigners = [
    galleryCurator,
    insuranceProvider,
    securityProvider
]
}

```

Example 2: Commodity Batch Management

```

-- Coffee batch tokenization example
commodityExample :: RealWorldAsset
commodityExample = RealWorldAsset {
    assetId = "coffee_batch_001",
    assetType = Commodity {
        type = "Arabica Coffee",
        grade = "Premium",
        origin = "Colombia",
        harvest = "2024",
        quantity = MetricTons 100
    },
    assetValue = 1_000_000_000_000, -- 1M ADA
    requirementSet = [
        Requirement {
            requirementType = Environmental,
            description = "Storage humidity 60-65%",
            validationMethod = HumiditySensorValidation,
            verificationFrequency = Daily,
            requirementStatus = Active
        }
    ]
}

-- Storage condition change proposal
proposeStorageChange :: ProposalParams

```

```

proposeStorageChange = ProposalParams {
  description = "Update storage conditions",
  effect = StorageConditionChange {
    newHumidity = Percentage 62,
    newTemperature = Celsius 20,
    implementation = "Automated climate control"
  },
  requiredCosigners = [
    warehouseManager,
    qualityInspector
  ]
}

```

3. Phase Transition Scenarios

Lock Phase Implementation

```

data LockPhaseInfo = LockPhaseInfo {
  startTime :: POSIXTime,
  endTime :: POSIXTime,
  requiredStake :: Integer,
  currentStake :: Integer,
  lockedTokens :: Map PubKeyHash Integer
}

startLockPhase :: ProposalId -> Contract w s Text ()
startLockPhase proposalId = do
  proposal <- getProposal proposalId
  currentTime <- getCurrentTime

  let lockInfo = LockPhaseInfo {
    startTime = currentTime,
    endTime = currentTime + proposal.lockDuration,
    requiredStake = calculateRequiredStake proposal,
    currentStake = 0,
    lockedTokens = Map.empty
  }

  validateLockPhaseStart proposal
  updateProposalPhase proposalId (LockPhase lockInfo)

```

```
emitLockPhaseStarted proposalId
```

```
validateLockPhaseStart :: Proposal -> Bool
```

```
validateLockPhaseStart proposal =
```

```
    hasMinimumCreationStake &&
```

```
    notInActivePhase &&
```

```
    allRequirementsValid
```

Voting Phase Implementation

```
data VotingPhaseInfo = VotingPhaseInfo {
```

```
    votes :: Map PubKeyHash Vote,
```

```
    totalVotingPower :: Integer,
```

```
    usedVotingPower :: Integer,
```

```
    voteDistribution :: VoteDistribution
```

```
}
```

```
data Vote = Vote {
```

```
    direction :: VoteDirection,
```

```
    power :: Integer,
```

```
    timestamp :: POSIXTime,
```

```
    metadata :: VoteMetadata
```

```
}
```

```
startVotingPhase :: ProposalId -> Contract w s Text ()
```

```
startVotingPhase proposalId = do
```

```
    proposal <- getProposal proposalId
```

```
-- Verify lock phase completion
```

```
unless (isLockPhaseComplete proposal) $
```

```
    throwError "Lock phase incomplete"
```

```
-- Verify start threshold
```

```
unless (meetsStartThreshold proposal) $
```

```
    throwError "Insufficient stake for voting start"
```

```
let votingInfo = VotingPhaseInfo {
```

```
    votes = Map.empty,
```

```
    totalVotingPower = calculateTotalPower proposal,
```

```
    usedVotingPower = 0,
```

```

        voteDistribution = initializeVoteDistribution
    }

    updateProposalPhase proposalId (VotingPhase votingInfo)
    emitVotingPhaseStarted proposalId

```

4. Threshold Calculations

Practical Examples

```

-- Example: Art piece worth 10M ADA
calculateThresholds :: AssetValue -> AgoraThresholds
calculateThresholds assetValue = AgoraThresholds {
    creation = Percentage 5,    -- 500k ADA to create proposal
    start = Percentage 10,     -- 1M ADA to start voting
    vote = Percentage 15,      -- 1.5M ADA for valid vote
    execution = Percentage 51, -- 5.1M ADA to execute
    cosigning = Percentage 25  -- 2.5M ADA for cosigning
}

-- Example: Calculating voting power
calculateVotingPower :: TokenAmount -> VotingStrategy -> Integer
calculateVotingPower amount strategy = case strategy of
    Linear ->
        amount
    Quadratic ->
        floor $ sqrt $ fromIntegral amount
    WeightedByTime holdingTime ->
        amount * (1 + (holdingTime `div` 30)) -- Bonus for longer holds

```

5. Governance Flows

Complete Proposal Lifecycle

```

data ProposalLifecycle = ProposalLifecycle {
    proposal :: Proposal,
    currentPhase :: AgoraPhase,
    history :: [PhaseTransition],
    votes :: Map PubKeyHash Vote,
    effects :: [Effect],

```

```

status :: ProposalStatus
}

executeProposalLifecycle :: ProposalId -> Contract w s Text ()
executeProposalLifecycle proposalId = do
  -- Initialize proposal
  proposal <- createProposal proposalId

  -- Lock phase
  startLockPhase proposalId
  awaitLockPhaseCompletion proposalId

  -- Voting phase
  startVotingPhase proposalId
  collectVotes proposalId

  -- Execution phase
  validateVotingResults proposalId
  collectCosignatures proposalId
  executeEffects proposalId

```

6. Implementation Considerations

Security Measures

```

data SecurityMeasures = SecurityMeasures {
  accessControl :: AccessControl,
  thresholdValidation :: ThresholdValidation,
  timelock :: TimelockConfig,
  emergencyProcedures :: EmergencyProcedures
}

validateSecurityMeasures :: SecurityMeasures -> ScriptContext -> Bool
validateSecurityMeasures measures ctx =
  validateAccess measures.accessControl ctx &&
  validateThresholds measures.thresholdValidation ctx &&
  validateTimelock measures.timelock ctx

```

Error Handling


```

data GovernanceError =
  InsufficientStake Text
| InvalidPhaseTransition Text
| ThresholdNotMet Text
| ValidationFailed Text
| TimelockNotExpired Text

handleGovernanceError :: GovernanceError -> Contract w s Text a
handleGovernanceError = \case
  InsufficientStake msg ->
    logError $ "Stake requirement not met: " <> msg
  InvalidPhaseTransition msg ->
    logError $ "Invalid phase transition: " <> msg
  ThresholdNotMet msg ->
    logError $ "Threshold not met: " <> msg
  ValidationFailed msg ->
    logError $ "Validation failed: " <> msg
  TimelockNotExpired msg ->
    logError $ "Timelock not expired: " <> msg

```

This implementation guide demonstrates how the Agora governance protocol can be used effectively with various real-world assets, providing both technical implementation details and practical examples that engineers can follow.

Revision #1
 Created 28 November 2024 00:42:13 by Aric Fedida
 Updated 28 November 2024 01:19:08 by Aric Fedida