# Cosmos代码分析

## 大体数据结构与说明

### baseapp/baseapp.go

mainConsensusParamsKey商店中共识参数存储的Key

type BaseApp struct {

// initialized on creation

logger log.Logger

name string // 来自abci.Info的应用程序名称

db dbm.DB // 常见DB后端

cms sdk.CommitMultiStore // 主要（未缓存）状态

router sdk.Router // 处理任何类型的消息

queryRouter sdk.QueryRouter // 用于重定向查询调用的路由器

txDecoder sdk.TxDecoder // []byte转换为sdk.Tx

baseKey \*sdk.KVStoreKey // cms中的主要KVStore 在LoadVersion或LoadLatestVersion上设置

anteHandler sdk.AnteHandler // 费用和认证的赌注处理程序

initChainer sdk.InitChainer // 使用验证器和状态blob初始化状态

beginBlocker sdk.BeginBlocker // 在任何tx之前运行的逻辑

endBlocker sdk.EndBlocker // 在所有tx之后运行的逻辑，并确定valset更改

addrPeerFilter sdk.PeerFilter // 按地址和端口过滤对等体

idPeerFilter sdk.PeerFilter // 按节点ID过滤对等体

fauxMerkleMode bool // 如果为true，IAVL MountStores使用MountStoresDB来模拟速度

// --------------------

//易失性状态

checkState \*state // for CheckTx 在初始化时设置并在Commit上重置

deliverState \*state // for DeliverTx 在InitChain和BeginBlock中设置并在Commit上清除

voteInfos []abci.VoteInfo // 来自开始块的缺失验证器

consensusParams \*abci.ConsensusParams // 共识参数 TODO：将来将此移动到主商店的baseapp param商店。

minGasPrices sdk.DecCoins // 验证者愿意接受处理交易的最低汽油价格。这主要用于DoS和垃圾邮件预防

sealed bool // 用于密封选项和BaseApp参数的标志

haltHeight uint64 // 停止链并正常关闭的高度

appVersion string // 应用程序的版本字符串

}

### simapp/app.go

// Extended ABCI application

type SimApp struct {

\*bam.BaseApp

cdc \*codec.Codec

invCheckPeriod uint

// keys to access the substores

keyMain \*sdk.KVStoreKey

keyAccount \*sdk.KVStoreKey

keyStaking \*sdk.KVStoreKey

tkeyStaking \*sdk.TransientStoreKey

keySlashing \*sdk.KVStoreKey

keyMint \*sdk.KVStoreKey

keyDistr \*sdk.KVStoreKey

tkeyDistr \*sdk.TransientStoreKey

keyGov \*sdk.KVStoreKey

keyFeeCollection \*sdk.KVStoreKey

keyParams \*sdk.KVStoreKey

tkeyParams \*sdk.TransientStoreKey

// keepers

accountKeeper auth.AccountKeeper

feeCollectionKeeper auth.FeeCollectionKeeper

bankKeeper bank.Keeper

stakingKeeper staking.Keeper

slashingKeeper slashing.Keeper

mintKeeper mint.Keeper

distrKeeper distr.Keeper

govKeeper gov.Keeper

crisisKeeper crisis.Keeper

paramsKeeper params.Keeper

// the module manager

mm \*module.Manager

}

### /client/context/context.go

// CLIContext实现了在SDK模块中创建的典型CLI上下文，用于事务处理和查询。

type CLIContext struct {

Codec \*codec.Codec

AccDecoder authtypes.AccountDecoder

Client rpcclient.Client

Keybase cryptokeys.Keybase

Output io.Writer

OutputFormat string

Height int64

NodeURI string

From string

AccountStore string

TrustNode bool

UseLedger bool

BroadcastMode string

PrintResponse bool

Verifier tmlite.Verifier

VerifierHome string

Simulate bool

GenerateOnly bool

FromAddress sdk.AccAddress

FromName string

Indent bool

SkipConfirm bool

}

### /store/cachkv/store.go

// 如果value为nil但删除为false，则表示父级没有密钥。 （无需在Write（）上删除）

type cValue struct {

value []byte

deleted bool

dirty bool

}

// Store围绕底层类型.KVStore包装内存缓存

type Store struct {

mtx sync.Mutex

cache map[string]\*cValue

unsortedCache map[string]struct{}

sortedCache \*list.List // always ascending sorted

parent types.KVStore

}

### /store/cachkv/mergeiterator.go

// cacheMergeIterator合并父Iterator和缓存Iterator。

// 缓存迭代器可以返回nil键来表示项目已被删除（但未在父级中删除）。

// 如果缓存迭代器具有与父级相同的密钥，则缓存阴影（覆盖）父级。

// TODO：通过记忆优化。

type cacheMergeIterator struct {

parent types.Iterator

cache types.Iterator

ascending bool

}

### /store/cachkv/memiterator.go

// 迭代iterKVCache项目。

// 如果key为nil，表示已删除。

// 实现迭代器。

type memIterator struct {

start, end []byte

items []\*cmn.KVPair

ascending bool

}

### /store/cachemulti/storei.go

// Store拥有许多缓存包装的商店。

//实现MultiStore。

//注意：商店（以及一般的MultiStores）不应该公开

子房的钥匙。

type Store struct {

db types.CacheKVStore

stores map[types.StoreKey]types.CacheWrap

keys map[string]types.StoreKey

traceWriter io.Writer

traceContext types.TraceContext

}

### /store/dbadapter/store.go

//dbm.Db的包装类型，实现了KVStore

type Store struct {

dbm.DB

}

## 整体分析

综合来看，整体项目中的数据结构与方法集成于/baseapp目录调用，其中/store为重要目录。具体代码内聚性较高，对应低耦合度，调用清晰，代码量分配平均。