# HARD\_FORK\_05更新修改

# 1. 功能调整

## 1.1 新增资产市场交易功能

新增市场交易功能，用于链上不同的资产之间的交易。由资产买卖双方发起限价订单，然后由底层撮合相互匹配的订单，撮合成功则立即执行，撮合失败则可以暂存在链上，以待其他的限价订单来撮合。资产交易需要扣除接收资产一部分作为市场交易手续费，该续费一部分返还给资产的手续费池，另一部分由交易账户的注册人和引荐人分红。具体划分比例由理事会参数决定。

其中资产交易手续费执行流程如下：

自定义资产创建的时候初始化market\_fee\_percent， 以及reward\_percent。其中market\_fee\_percent为资产交易手续费比例，reward\_percent为资产交易手续费中给予账户推荐者的奖励比例。两个参数都可以被资产所有人，使用asset\_update\_operation来修改。

计算流程：

计算交易手续费： pay = amount \* market\_fee\_percent

计算手续费的推荐奖励：award = pay \* reward\_percent

计算返还给资产手续费池：fees += pay – award

计算引荐人奖励：referrer\_award = award \* referrer\_percent

计算注册者奖励：registrar\_award = award – referrer\_award

registrar\_percent

账户推荐奖励award

注册者奖励

registrar\_awarddd

引荐人奖励

referrer\_award

market\_fee\_percent

reward\_percent

1 - reward\_percent

referrer\_percent

## 1.2 新增锁仓功能

新增锁仓功能，账户可以锁定余额，用于产生积分（HARD\_FORK\_05之后，积分产生方式只与锁仓有关），可以取消锁仓，余额7天（理事会参数，可调整）之后返还给账户。

## 1.3 积分产生方式调整

积分产生公式不变，其中的有效余额调整，具体如下：

原有效余额 =（账户余额 + 借入用于产生积分的额度 – 借出用于产生积分的额度 – 用于见证人抵押的额度）

**effective\_balance = core\_balance + core\_leased\_in - core\_leased\_out - total\_witness\_pledge**

调整为 账户需要锁定一部分YOYO资产替代有效余额。

新有效余额 = 锁定的用于产生积分的额度

**effective\_balance = locked\_balance\_for\_feepoint**

调整之后，账户的积分累积只与锁定的YOYO以及时间有关，不再是账户所有的有效余额参与产生积分。locked\_balance\_for\_feepoint具体值，为账户的Lock\_balance类型的抵押pledge\_balance\_object中的总抵押pledge。

## 1.4 新增抵押挖矿功能

普通账户可以参与抵押挖矿，账户抵押资产(YOYO)给见证人，有最小抵押限制（理事会参数控制，默认是1000YOYO）；见证人参与抵押出块，获取的出块奖励，分成两部分，一部分见证人收取，一部分作为抵押分红按抵押量均分给抵押账户，两部分收益比例由抵押分红比例（见证人设定）确定，假设出块工资为1YOYO，抵押分红比例为90%，则见证人获取0.1YOYO，其他抵押账户均分0.9YOYO；只有见证人开启挖矿之后，账户才能抵押给该见证人，见证人不能抵押给自己，账户可以抵押给多个见证人，可以更新抵押，更新抵押金额也不能少于最小抵押，可以取消抵押，取消抵押时，不立刻返还抵押金额给账户，有默认7天（理事会参数控制）的锁定期，到期后自动返还给抵押账户；如果见证人取消了自己的见证人抵押（即删除见证人），那么所有抵押金额7天（理事会参数控制）之后返还给抵押账户。

## 1.5 更新注册账户

HARD\_FORK\_05之后，注册账户时注册商与引荐人不能为空账户，并且注册商与引荐人分红比率之和必须等于100%。

## 1.6 注册商、引荐人修改

由于要引入注册商、引荐人点赞分红功能，在HARD\_FORK\_05之前注册的账户需要在分叉点修改：

1. 如果注册商为空账户，则注册商账号修改为224373708；如果引荐人为空账户，则引荐人账户修改为23080。
2. 注册商与引荐人分红比例都修改为50%

## 1.7 新增注册商、引荐人点赞分红功能

HARD\_FORK\_05之后，账户点赞之后获得的收益，一部分要分给注册商与引荐人，这部分比例由理事会参数（registrar\_referrer\_rate\_from\_score）控制，默认是25%；

点赞分红按按注册商分红比例与引荐人分红比例分发，注册商比例与引荐人比例之和为100%。

## 1.8 代码优化调整

新增pledge\_balance\_object用于记录和处理资产抵押，以及在\_account\_statistics\_object里面添加map<pledge\_balance\_type,pledge\_balance\_id\_type>用于处理该账户所有的资产抵押。优化去掉原account\_statistics\_object中total\_witness\_pledge， releasing\_witness\_pledge等相似变量。

## 1.9 资产抵押释放调整

在hard\_fork05之前，资产抵押新的释放块号会覆盖旧的释放块号，例如平台抵押释放为n个块号时，新的释放将覆盖旧的块号，即用head\_block\_num + GRAPHENE\_DEFAULT\_PLATFORM\_PLEDGE\_RELEASE\_DELAY 来替换块号n。

在hard\_fork05之后，抵押资产的释放改为分批次释放，新的释放块号不再覆盖原来的块号。在1.8的代码调整基础上，由pledge\_balance\_object中的map<uint32\_t,share\_type> releasing\_pledges来表示分批待释放抵押的表。Map中的key表示待释放的块号，value表示待释放金额。该资产释放表的大小由理事会参数max\_pledge\_releasing\_size决定，默认为20.

## 1.10 理事会发起提案撤回平台抵押

由于平台可以控制授权给该平台的所有零钱，如果平台存在作恶，将用户授权给该平台的零钱转走，理事会可以发起提案，撤回其平台抵押，惩罚作恶平台；如果理事会设置撤回金额大于平台的全部抵押与待释放抵押之后，则撤回金额为平台的全部抵押与待释放抵押之和；撤回抵押优先从待释放的抵押之中扣除；如果撤走部分抵押之后，并且剩余抵押小于平台最低抵押，那么删除平台对象，但是平台授权对象仍然存在。

# 2. 理事会参数调整

1. uint64\_t    min\_mining\_pledge                   = GRAPHENE\_DEFAULT\_MIN\_MINING\_PLEDGE;
2. uint32\_t    mining\_pledge\_release\_delay         = GRAPHENE\_DEFAULT\_MINING\_PLEDGE\_RELEASE\_DELAY;
3. uint32\_t    max\_pledge\_mining\_bonus\_rate        = GRAPHENE\_DEFAULT\_MAX\_PLEDGE\_MINING\_BONUS\_RATE;
4. uint32\_t    registrar\_referrer\_rate\_from\_score  = GRAPHENE\_DEFAULT\_REGISTRAR\_REFERRER\_RATE\_FROM\_SCORE;
5. uint32\_t    unlocked\_balance\_release\_delay      = GRAPHENE\_DEFAULT\_UNLOCKED\_BALANCE\_RELEASE\_DELAY;
6. uint32\_t    max\_pledge\_releasing\_size           = GRAPHENE\_DEFAULT\_MAX\_PLEDGE\_RELEASING\_SIZE;
7. uint32\_t    scorer\_earnings\_rate                = GRAPHENE\_DEFAULT\_SCORER\_EARNINGS\_RATE;

## 2.1 新增参数unlocked\_balance\_release\_delay

该参数为账户调整锁定产生积分的额度的时候，待释放的额度会在经过unlocked\_balance\_release\_delay块后，将待释放额度返还给账户。该参数默认值为uint32\_t(28800\*7)，即在七天后返回待释放额度。

## 2.2 新增参数min\_mining\_pledge

账户抵押给见证人挖矿最小抵押额度，低于该值抵押操作会失败，默认值为1000YOYO

## 2.3 新增参数mining\_pledge\_release\_delay

账户取消抵押挖矿，抵押金额有一个释放延时（以出块数量计算），当出块数量超过该值之后，抵押金额才能返还给账户，释放延时默认为28800\*7块，即7天。

## 2.4 新增参数max\_pledge\_mining\_bonus\_rate

见证人可以设置抵押挖矿分红比例，但是抵押分红比例不能超过该值，默认为9000（即90%），更新范围为10%-90%；

抵押挖矿分红比例：见证人出块工资的一部分分成给按抵押数量均分给抵押账户，假设出块工资为1YOYO，分红比例为90%，则见证人获取0.1YOYO，其他抵押账户均分0.9YOYO。

## 2.5 新增参数registrar\_referrer\_rate\_from\_score

注册商与引荐人点赞分红比例，点赞获取的收益按比例分成给注册商与引荐人，默认为2500（25%），更新范围为0-25%。

## 2.6 新增参数max\_pledge\_releasing\_size

资产抵押分批次释放的最大数量，默认为20, 更新范围必须大于等于1；

1. **hardfork\_05**之前，分批次释放数量固定为1；

2. **hardfork\_05** 及之后，分批次释放数量为该值；

## 2.7 新增参数scorer\_earnings\_rate

点赞收入与文章收益的占比，文章收益的收益按比例分配给点赞人。默认为2500，即25%，更新范围为大于等于0，小于100%；

1. **hardfork\_04** 时固定25%；

2. **hardfork\_05**及之后点赞收入与文章收益的占比由该参数控制；

# 3. 插件改动整理

## 3.1 新增插件market\_history\_plugin

新增资产交易市场插件功能，用于记录查询资产交易详情。包括限价订单列表，交易价格查询、交易历史查询等等。

# 4. 数据结构整理

## 4.1 新增limit\_order\_object

1. **class** limit\_order\_object : **public** abstract\_object<limit\_order\_object>
2. {
3. **public**:
4. **static** **const** uint8\_t space\_id = protocol\_ids;
5. **static** **const** uint8\_t type\_id  = limit\_order\_object\_type;
7. time\_point\_sec   expiration;   //订单过期时间
8. account\_uid\_type seller;       //订单出售者
9. share\_type       for\_sale;     //出售资产的数量
10. price            sell\_price;   //出售资产和接收资产的比价
11. };

更新说明：新增限价订单结构，用于存储限价订单，在资产交易市场中，匹配买卖双方，撮合资产交易订单完成。

## 4.2 新增limit\_order\_create\_operation

1. **struct** limit\_order\_create\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = GRAPHENE\_BLOCKCHAIN\_PRECISION / 10;
6. uint32\_t price\_per\_kbyte = 0 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
7. uint64\_t min\_real\_fee = 0;
8. uint16\_t min\_rf\_percent = 0;
9. extensions\_type   extensions;
10. };
12. fee\_type         fee;                                     //手续费
13. account\_uid\_type seller;                                  //限价订单出售者
14. asset            amount\_to\_sell;                          //出售的资产
15. asset            min\_to\_receive;                          //最低接收的资产
17. time\_point\_sec expiration = time\_point\_sec::maximum();    //订单过期时间
19. **bool** fill\_or\_kill = **false**;                                //true：若没匹配上则取消；false：没匹配上暂存在链上
20. extensions\_type   extensions;                             //扩展项
22. };

更新说明：新增创建限价订单操作。资产市场交易由买卖双方发起限价订单，然后由底层撮合匹配的限价订单。其中，fill\_or\_kill为true表示撮合成功则执行否则取消，为false表示撮合不成功则暂存在链上，在过期时间内等待其他限价订单撮合。

## 4.3 新增limit\_order\_cancel\_operation

1. **struct** limit\_order\_cancel\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = GRAPHENE\_BLOCKCHAIN\_PRECISION / 100;
6. uint32\_t price\_per\_kbyte = 0 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
7. uint64\_t min\_real\_fee = 0;
8. uint16\_t min\_rf\_percent = 0;
9. extensions\_type   extensions;
10. };
12. fee\_type             fee;                //手续费
13. limit\_order\_id\_type  order;              //要取消的订单id
14. account\_uid\_type     fee\_paying\_account; //订单的发起账户
15. extensions\_type      extensions;         //扩展项
16. };

更新说明：新增取消限价订单操作。由限价订单发起者，发起取消操作，用于取消在等待中未完成的限价订单。

## 4.4 新增market\_fee\_collect\_operation

1. **struct** market\_fee\_collect\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = 1 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
6. uint32\_t price\_per\_kbyte = 0 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
7. uint64\_t min\_real\_fee = 0;
8. uint16\_t min\_rf\_percent = 0;
9. extensions\_type   extensions;
10. };
12. fee\_type          fee;           //手续费
13. account\_uid\_type  account;       //发起账户
14. asset\_aid\_type    asset\_aid;     //资产的aid
15. share\_type        amount;        //要取回的资产数量
16. extensions\_type   extensions;    //扩展项
17. };

更新说明：新增收取资产市场交易手续费的收益。资产交易者账户的注册者和引荐人可以从该账户市场交易扣除的手续费中获取部分收益。该操作用于取回市场交易手续费产生的收益。

## 4.5 新增pledge\_mining\_object

1. **class** pledge\_mining\_object : **public** graphene::db::abstract\_object<pledge\_mining\_object>
2. {
3. **public**:
4. **static** **const** uint8\_t space\_id = implementation\_ids;
5. **static** **const** uint8\_t type\_id = impl\_pledge\_mining\_object\_type;
7. account\_uid\_type     pledge\_account;                                       //抵押挖矿账户
8. account\_uid\_type     witness;                                              //抵押挖矿见证人
9. share\_type           pledge;                                               //抵押数量
11. uint32\_t             last\_bonus\_block\_num = 0;                             //上一次抵押分红处理块号
12. share\_type           releasing\_mining\_pledge;                              //待释放的抵押
13. uint32\_t             mining\_pledge\_release\_block\_number = -1;              //抵押释放块号
14. };

更新说明：新增抵押挖矿结构， 普通账户抵押给见证人，从见证人出块工资中分红。

## 4.6 新增pledge\_mining\_update\_operation

1. **struct** pledge\_mining\_update\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = 1 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
6. uint64\_t min\_real\_fee = 0;
7. uint16\_t min\_rf\_percent = 0;
8. extensions\_type   extensions;
9. };
11. fee\_type            fee;               //手续费
12. account\_uid\_type    pledge\_account;    //抵押挖矿账户
13. account\_uid\_type    witness;           //抵押挖矿见证人
14. share\_type          new\_pledge;        //更新抵押量
16. extensions\_type     extensions;        //扩展项
17. };

更新说明：新增挖矿抵押更新操作，包含创建抵押挖矿，更新抵押挖矿，取消抵押挖矿 ；

限制条件：

1. 见证人开放抵押挖矿权限
2. 创建，更新抵押挖矿时抵押量必须不小于1000YOYO（理事会参数，可调整）
3. 见证人不能抵押给自己

## 4.7 新增pledge\_bonus\_collect\_operation

1. **struct** pledge\_bonus\_collect\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = 1 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
6. uint64\_t min\_real\_fee = 0;
7. uint16\_t min\_rf\_percent = 0;
8. extensions\_type   extensions;
9. };
11. fee\_type          fee;           //手续费
12. account\_uid\_type  account;       //抵押分红领取账户
13. share\_type        bonus;         //领取数量
14. extensions\_type   extensions;    //扩展项
15. };

更新说明：新增领取抵押分红操作，抵押挖矿分红并不是立即分发给账户，存储在每个账户的分红池中，可通过该操作领取到账户余额中。

## 4.8 新增score\_bonus\_collect\_operation

1. **struct** score\_bonus\_collect\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = 1 \* GRAPHENE\_BLOCKCHAIN\_PRECISION;
6. uint64\_t min\_real\_fee = 0;
7. uint16\_t min\_rf\_percent = 0;
8. extensions\_type   extensions;
9. };
11. fee\_type          fee;           //手续费
12. account\_uid\_type  account;       //抵押分红领取账户
13. share\_type        bonus;         //领取数量
14. extensions\_type   extensions;    //扩展项
15. };

更新说明：新增领取点赞分红操作，点赞获取收益的部分（比例由理事会参数确定） 按比例分配给引荐人与注册商，发放到各自的点赞分红池中，可通过该操作领取到账户余额中。

## 4.9 新增balance\_lock\_update\_operation

1. **struct** balance\_lock\_update\_operation : **public** base\_operation
2. {
3. **struct** fee\_parameters\_type
4. {
5. uint64\_t fee = GRAPHENE\_BLOCKCHAIN\_PRECISION / 10;
6. uint64\_t min\_real\_fee = 0;
7. uint16\_t min\_rf\_percent = 0;
8. extensions\_type   extensions;
9. };
11. fee\_type          fee;              //交易的手续费
12. account\_uid\_type  account;          //发起账户的uid
13. share\_type        new\_lock\_balance; //新的用于产生积分锁定额度
14. extensions\_type   extensions;       //扩展项
15. };

更新说明：新增balance\_lock\_update\_operation用于抵押YOYO产生积分。其中new\_lock\_balance为新的锁定额度。新的额度每次执行都是及时生效，新的额度与就额度的差值记录在accout\_statistics\_object的releasing\_locked\_feepoint中作为待释放额度。在经过理事会参数中unlocked\_balance\_release\_delay个块后，待释放额度返回账户。

## 4.10 修改witness\_object

1. **class** witness\_object : **public** abstract\_object<witness\_object>
2. {
3. **public**:
4. **static** **const** uint8\_t space\_id = protocol\_ids;
5. **static** **const** uint8\_t type\_id = witness\_object\_type;
7. account\_uid\_type    account;
8. string              name;
9. uint32\_t            sequence;
10. **bool**                is\_valid = **true**;
12. public\_key\_type     signing\_key;
14. uint64\_t            pledge;
15. fc::time\_point\_sec  pledge\_last\_update;
16. uint64\_t            average\_pledge = 0;
17. fc::time\_point\_sec  average\_pledge\_last\_update;
18. uint32\_t            average\_pledge\_next\_update\_block;
20. uint64\_t            total\_votes = 0;
22. fc::uint128\_t       by\_pledge\_position;
23. fc::uint128\_t       by\_pledge\_position\_last\_update;
24. fc::uint128\_t       by\_pledge\_scheduled\_time = fc::uint128\_t::max\_value();
26. fc::uint128\_t       by\_vote\_position;
27. fc::uint128\_t       by\_vote\_position\_last\_update;
28. fc::uint128\_t       by\_vote\_scheduled\_time = fc::uint128\_t::max\_value();
30. uint32\_t            last\_confirmed\_block\_num = 0;
32. uint64\_t            last\_aslot = 0;
33. uint64\_t            total\_produced = 0;
34. uint64\_t            total\_missed = 0;
35. string              url;
37. **bool**                can\_pledge = **false**;         //true,开启抵押挖矿功能;false,关闭
38. uint32\_t            bonus\_rate = 0;             //抵押挖矿分红比例，出块工资中的该比例部分分给抵押账户
39. uint64\_t            total\_mining\_pledge = 0;    //抵押给该见证人的所有抵押
40. map<uint32\_t, share\_type> bonus\_per\_pledge;     //记录每个时间段的每个抵押的分红金额，key为块号，value为每个抵押分红金额；如果该见证人的总挖矿抵押变更，记录一次
41. share\_type          unhandled\_bonus;            //该见证人的总抵押距离上一次变更之时，这个时间段所有的分红，即没有计算bonus\_per\_pledge的数量
42. share\_type          need\_distribute\_bonus;      //需要给抵押者的分红
43. share\_type          already\_distribute\_bonus;   //已经分发给抵押者的分红
44. uint32\_t            last\_update\_bonus\_block\_num = 0; //该见证人上一次分红块号，间隔为10000块
46. };

更新说明：

此部分更新主要用于计算与发放抵押挖矿分红，见证人出块工资按照挖矿分红比例（bonus\_rate）累加到未处理的分红（unhandled\_bonus）中，当有账户创建、更新或取消该见证人的挖矿抵押时，根据bonus\_per\_pledge与该账户抵押给该见证人的金额计算该账户的分红并发放到该账户的抵押挖矿奖金池里，同时计算该时间段的每抵押分红，更新bonus\_per\_pledge, unhandled\_bonus清零，然后全局每出一万个块（从last\_update\_bonus\_block\_num算起），就对该见证人的所有抵押账户发放分红奖励；由于计算过程中的除法余数问题，导致实际发放的分红数与需要发放的分红存在稍许偏差，所以用need\_distribute\_bonus和already\_distribute\_bonus记录下来，等到对所有见证人发放奖励的时候，

need\_distribute\_bonus减去already\_distribute\_bonus的就是偏差部分，将该部分发放到见证人出块工资里。

## 4.11 修改类account\_statistics\_object为\_account\_statistics\_object

1. **class** \_account\_statistics\_object : **public** graphene::db::abstract\_object<\_account\_statistics\_object>
2. {
3. **public**:
4. **static** **const** uint8\_t space\_id = implementation\_ids;
5. **static** **const** uint8\_t type\_id  = impl\_account\_statistics\_object\_type;
7. account\_uid\_type  owner;
9. account\_transaction\_history\_id\_type most\_recent\_op;
10. uint32\_t                            total\_ops = 0;
11. uint32\_t                            removed\_ops = 0;
13. share\_type total\_core\_in\_orders; // add for limit order
14. share\_type prepaid;
15. share\_type csaf;
17. share\_type core\_balance;
18. share\_type core\_leased\_in;
19. share\_type core\_leased\_out;
21. share\_type                     average\_coins;
22. fc::time\_point\_sec             average\_coins\_last\_update;
24. fc::uint128\_t                  coin\_seconds\_earned;
25. fc::time\_point\_sec             coin\_seconds\_earned\_last\_update;
27. share\_type total\_mining\_pledge=0;
29. uint32\_t last\_witness\_sequence = 0;
31. share\_type uncollected\_witness\_pay;
32. share\_type uncollected\_pledge\_bonus;
33. map<asset\_aid\_type, share\_type> uncollected\_market\_fees;
34. share\_type uncollected\_score\_bonus;
36. uint64\_t witness\_last\_confirmed\_block\_num = 0;
37. uint64\_t witness\_last\_aslot = 0;
38. uint64\_t witness\_total\_produced = 0;
39. uint64\_t witness\_total\_missed = 0;
40. uint64\_t witness\_last\_reported\_block\_num = 0;
41. uint64\_t witness\_total\_reported = 0;
43. uint32\_t last\_committee\_member\_sequence = 0;
45. **bool** can\_vote = **true**;
46. **bool** is\_voter = **false**;
48. uint32\_t last\_voter\_sequence = 0;
50. uint32\_t last\_platform\_sequence = 0;
52. post\_pid\_type last\_post\_sequence = 0;
54. custom\_vote\_vid\_type last\_custom\_vote\_sequence = 0;
55. advertising\_aid\_type last\_advertising\_sequence = 0;
56. license\_lid\_type     last\_license\_sequence = 0;

59. **template**<**class** DB>
60. share\_type get\_all\_pledge\_balance(asset\_aid\_type asset\_id,**const** DB& db)**const**{
61. share\_type res=0;
62. **for**(**const** auto & type\_id:pledge\_balance\_ids){
63. auto pledge\_balance\_obj = db.get(type\_id.second);
64. **if**(pledge\_balance\_obj.asset\_id==asset\_id)
65. res+=pledge\_balance\_obj.total\_unrelease\_pledge();
66. }
67. **return** res;
68. }
69. **template**<**class** DB>
70. share\_type get\_pledge\_balance(asset\_aid\_type asset\_id,pledge\_balance\_type type,**const** DB& db)**const**{
71. **if**(pledge\_balance\_ids.count(type)!=0){
72. pledge\_balance\_object pledge\_balance\_obj=db.get(pledge\_balance\_ids.at(type));
73. **if**(pledge\_balance\_obj.asset\_id==asset\_id)
74. **return** pledge\_balance\_obj.total\_unrelease\_pledge();
75. }
76. **return**  0;
77. }
79. **template**<**class** DB>
80. share\_type get\_releasing\_pledge(asset\_aid\_type asset\_id, pledge\_balance\_type type, **const** DB& db) **const** {
81. **if** (pledge\_balance\_ids.count(type) != 0){
82. auto pledge\_balance\_obj = db.get(pledge\_balance\_ids.at(type));
83. **if** (pledge\_balance\_obj.asset\_id == asset\_id)
84. **return** pledge\_balance\_obj.total\_releasing\_pledge;
85. }
86. **return**  0;
87. }
89. **template**<**class** DB>
90. share\_type get\_available\_core\_balance(**const** DB& db) **const**{
91. **return** core\_balance - get\_all\_pledge\_balance(GRAPHENE\_CORE\_ASSET\_AID, db)
92. - total\_mining\_pledge
93. - core\_leased\_out;
94. }
96. **template**<**class** DB>
97. share\_type get\_available\_core\_balance(pledge\_balance\_type exclude\_type, **const** DB& db) **const**{
98. **return** get\_available\_core\_balance(db) +
99. get\_pledge\_balance(GRAPHENE\_CORE\_ASSET\_AID, exclude\_type, db);
100. }
102. map<pledge\_balance\_type,pledge\_balance\_id\_type> pledge\_balance\_ids; //新增账户抵押资产表
104. };

更新说明：新增账户抵押资产表，用于统一处理处理和记录见证人，理事会，平台，锁定积分，抵押挖矿等相关资产抵押。删除了原来结构的total\_witness\_pledge， releasing\_witness\_pledge，witness\_pledge\_release\_block\_number，locked\_balance\_for\_feepoint，releasing\_locked\_feepoint，feepoint\_unlock\_block\_number，total\_mining\_pledge。

## 4.12 新增结构体account\_statistics\_object

1. **class** account\_statistics\_object : **public** graphene::db::abstract\_object<account\_statistics\_object>
2. {
3. **public**:
4. **static** **const** uint8\_t space\_id = implementation\_ids;
5. **static** **const** uint8\_t type\_id  = impl\_account\_statistics\_object\_type;
7. account\_uid\_type  owner;
9. account\_transaction\_history\_id\_type most\_recent\_op;
10. uint32\_t                            total\_ops = 0;
11. uint32\_t                            removed\_ops = 0;
13. share\_type prepaid;
14. share\_type csaf;
15. share\_type core\_balance;
16. share\_type core\_leased\_in;
17. share\_type core\_leased\_out;
19. share\_type                     average\_coins;
20. fc::time\_point\_sec             average\_coins\_last\_update;
21. fc::uint128\_t                  coin\_seconds\_earned;
22. fc::time\_point\_sec             coin\_seconds\_earned\_last\_update;
24. share\_type total\_witness\_pledge;
25. share\_type releasing\_witness\_pledge;
26. uint32\_t   witness\_pledge\_release\_block\_number = -1;
28. share\_type locked\_balance\_for\_feepoint;       //锁定的额度，用于产生积分
29. share\_type releasing\_locked\_feepoint;         //待释放的额度
30. uint32\_t   feepoint\_unlock\_block\_number = -1; //下一次返还待释放额度的块号
31. share\_type total\_mining\_pledge=0;  //抵押给见证人的总挖矿抵押
32. share\_type uncollected\_pledge\_bonus; //未收集的挖矿抵押分红
33. share\_type uncollected\_score\_bonus; //注册商、引荐人未收集的点赞分红
34. map<asset\_aid\_type, share\_type> uncollected\_market\_fees; //未收集的资产交易市场费用
36. uint32\_t   last\_witness\_sequence = 0;
37. share\_type uncollected\_witness\_pay;
39. uint64\_t witness\_last\_confirmed\_block\_num = 0;
40. uint64\_t witness\_last\_aslot = 0;
41. uint64\_t witness\_total\_produced = 0;
42. uint64\_t witness\_total\_missed = 0;
43. uint64\_t witness\_last\_reported\_block\_num = 0;
44. uint64\_t witness\_total\_reported = 0;
46. share\_type total\_committee\_member\_pledge;
47. share\_type releasing\_committee\_member\_pledge;
48. uint32\_t   committee\_member\_pledge\_release\_block\_number = -1;
49. uint32\_t   last\_committee\_member\_sequence = 0;
51. **bool**     can\_vote = **true**;
52. **bool**     is\_voter = **false**;
53. uint32\_t last\_voter\_sequence = 0;
54. uint32\_t last\_platform\_sequence = 0;
56. share\_type    total\_platform\_pledge;
57. share\_type    releasing\_platform\_pledge;
58. uint32\_t      platform\_pledge\_release\_block\_number = -1;
59. post\_pid\_type last\_post\_sequence = 0;
61. custom\_vote\_vid\_type last\_custom\_vote\_sequence = 0;
62. advertising\_aid\_type last\_advertising\_sequence = 0;
63. license\_lid\_type     last\_license\_sequence = 0;
64. };

更新说明：为了不改变get\_full\_accounts的返回结果，新增一个account\_statistics\_object结构体，用于返回账户统计结果。原account\_statistics\_object结构改为\_account\_statistics\_object.

## 4.13 新增枚举pledge\_balance\_type

1. **enum** pledge\_balance\_type{
2. Witness,      //见证人抵押
3. Commitment,   //理事会抵押
4. Platform,     //平台抵押
5. Lock\_balance, //积分抵押
6. Mine          //挖矿抵押
7. };

更新说明：新增枚举结构pledge\_balance\_type，用于表示用户抵押资产的类型。

## 4.14 新增结构体pledge\_balance\_object

1. **class** pledge\_balance\_object:**public** graphene::db::abstract\_object<pledge\_balance\_object>{
2. **public**:
3. **static** **const** uint8\_t space\_id = implementation\_ids;
4. **static** **const** uint8\_t type\_id  = impl\_pledge\_balance\_object\_type;
6. uint64\_t             superior\_index;                     //复用的索引，在抵押挖矿的时候表示pledge\_mining\_object的id.instance()，其他情况下表示抵押人的账户id
7. pledge\_balance\_type  type;                               //资产抵押类型
8. asset\_aid\_type       asset\_id = GRAPHENE\_CORE\_ASSET\_AID; //资产符号
9. share\_type           pledge;                             //已抵押资产的数量
10. share\_type           total\_releasing\_pledge=0;           //待释放资产总量
11. map<uint32\_t,share\_type> releasing\_pledges;              //待释放资产详情表，map的key：待释放的块号，value：释放的数量
13. uint64\_t earliest\_release\_block\_number()**const**{
14. **if**(releasing\_pledges.empty())
15. **return** uint32\_t(-1);
16. **else**
17. **return** releasing\_pledges.begin()->first;
18. }
20. uint64\_t last\_release\_block\_number()**const**{
21. **if** (releasing\_pledges.empty())
22. **return** uint32\_t(-1);
23. **else**
24. **return** releasing\_pledges.rbegin()->first;
25. }
27. //return the delta pledge need to subtract from account balance
28. **template**< **typename** DB>
29. share\_type update\_pledge(**const** asset &new\_pledge, uint32\_t new\_relase\_num, **const** DB &db){
31. FC\_ASSERT(new\_pledge.asset\_id == asset\_id, "erro asset id ");
32. auto delta = new\_pledge.amount - total\_unrelease\_pledge();
33. auto delta\_releasing = pledge - new\_pledge.amount;
34. pledge = new\_pledge.amount;
36. **if** (delta >= 0) {
37. releasing\_pledges.clear();
38. total\_releasing\_pledge=0;
39. **return** delta;
40. } **else** {
41. **if** (delta\_releasing > 0)
42. new\_releasing(delta\_releasing, new\_relase\_num, db);
43. **else**
44. reduce\_releasing(-delta\_releasing);
45. **return** 0;
46. }
47. }
49. **template**< **typename** DB>
50. **void** new\_releasing(**const** asset &new\_releasing\_pledge, uint32\_t new\_relase\_num, **const** DB &db){
52. FC\_ASSERT(new\_releasing\_pledge.asset\_id == asset\_id, "erro asset id ");
53. uint16\_t max\_releasing\_size = 0;
54. **const** auto& dpo = db.get\_dynamic\_global\_properties();
55. **if** (dpo.enabled\_hardfork\_version < ENABLE\_HEAD\_FORK\_05)
56. max\_releasing\_size = 1;
57. **else** {
58. **const** auto& global\_params = db.get\_global\_properties().parameters;
59. max\_releasing\_size = global\_params.get\_award\_params().max\_pledge\_releasing\_size;
60. }
61. **if** (releasing\_pledges.size() == max\_releasing\_size)
62. {
63. auto iter = --releasing\_pledges.end();
64. auto last\_releasing\_pledge = iter->second;
65. releasing\_pledges.erase(iter);
66. releasing\_pledges[new\_relase\_num] = last\_releasing\_pledge + new\_releasing\_pledge.amount;
67. }
68. **else** **if** (releasing\_pledges.size() < max\_releasing\_size)
69. releasing\_pledges[new\_relase\_num] += new\_releasing\_pledge.amount;
70. **else**
71. FC\_ASSERT(**false**, "releasing pledge size is invaild");
72. total\_releasing\_pledge += new\_releasing\_pledge.amount;
74. }
75. share\_type total\_unrelease\_pledge()**const** { **return** pledge + total\_releasing\_pledge; }
77. **void** reduce\_releasing(share\_type amount) {
78. FC\_ASSERT(total\_releasing\_pledge >= amount, " available releasing balance is not enough");
79. total\_releasing\_pledge -= amount;
80. uint32\_t erase\_from = -1;
81. **for** (auto itr = releasing\_pledges.rbegin(); itr != releasing\_pledges.rend(); ++itr) {
82. **if** (itr->second <= amount) {
83. amount -= itr->second;
84. erase\_from = itr->first;
85. } **else**{
86. releasing\_pledges[itr->first] -= amount;
87. **break**;
88. }
89. }
90. **if** (erase\_from != uint32\_t(-1))
91. releasing\_pledges.erase(releasing\_pledges.lower\_bound(erase\_from), releasing\_pledges.end());
92. }
93. };

更新说明：新增pledge\_balance\_object，用于统一记录和处理账户的资产抵押，包括见证人，理事会，平台，锁定积分，抵押挖矿等。几种资产抵押由该结构来完成，包括创建，更新和释放资产抵押。

## 4.15 修改post\_object

1. **class** post\_object : **public** graphene::db::abstract\_object<post\_object>
2. {
3. **public**:
4. **enum** Post\_Permission
5. {
6. Post\_Permission\_Forward = 1,   //allow forward
7. Post\_Permission\_Liked   = 2,   //allow liked or scored
8. Post\_Permission\_Buyout  = 4,   //allow buyout
9. Post\_Permission\_Comment = 8,   //allow comment
10. Post\_Permission\_Reward  = 16   //allow reward
11. };
13. **static** **const** uint8\_t space\_id = protocol\_ids;
14. **static** **const** uint8\_t type\_id  = post\_object\_type;
16. /// The platform's pid.
17. account\_uid\_type             platform;
18. /// The poster's uid.
19. account\_uid\_type             poster;
20. /// The post's pid.
21. post\_pid\_type                post\_pid;
22. /// The post's type.
23. post\_operation::Post\_Type    post\_type = post\_operation::Post\_Type::Post\_Type\_Post; //文章类型
24. /// If it is a transcript, this value is requested as the source author uid
25. optional<account\_uid\_type>   origin\_poster;
26. /// If it is a transcript, this value is required for the source id
27. optional<post\_pid\_type>      origin\_post\_pid;
28. /// If it is a transcript, this value is required for the source platform
29. optional<account\_uid\_type>   origin\_platform;
31. string                       hash\_value;
32. string                       extra\_data; ///< category, tags and etc
33. string                       title;
34. string                       body;
36. time\_point\_sec create\_time;
37. time\_point\_sec last\_update\_time;
39. map<account\_uid\_type, Receiptor\_Parameter> receiptors; //receiptors of the post
40. optional<share\_type>                       forward\_price;
41. optional<license\_lid\_type>                 license\_lid;
42. uint32\_t                                   permission\_flags = 0xFF;
43. **bool**                                       score\_settlement = **false**;
44. };

更新说明：新增post\_operation::Post\_Type，用于记录文章类型，区分文章是发文，评论，转发，转发与修改；hardfork\_04之前的文章统一默认为发文；

## 4.16 修改asset\_options结构

1. **struct** additional\_asset\_options
2. {
3. optional<uint16\_t>   reward\_percent;
4. };
5. **typedef** extension<additional\_asset\_options> additional\_asset\_options\_t;

asset\_options的扩展结构改为了optional<additional\_asset\_options\_t> extensions，添加了资产交易手续费的引荐人奖励比例，详情见上文资产交易手续费执行流程。

## 4.17 新增beneficiary\_assign\_operation

1. **struct** beneficiary\_assign\_operation : **public** base\_operation
2. {
3. fee\_type         fee;             //手续费
4. account\_uid\_type owner;           //账户
5. account\_uid\_type new\_beneficiary; //新的受益人
7. extensions\_type  extensions;      //扩展参数
8. };

新增beneficiary\_assign\_operation用于指定账户的受益人，收益的类型为见证人工资或锁仓产生的积分。

## 4.18 新增benefit\_collect\_operation

1. **struct** benefit\_collect\_operation : **public** base\_operation
2. {
3. **enum** BENEFIT\_TYPE{
4. BENEFIT\_TYPE\_CSAF = 0,    //锁仓积分
5. BENEFIT\_TYPE\_WITNESS = 1  //见证人工资
6. };
8. fee\_type                   fee;          //手续费
9. account\_uid\_type           issuer;       //受益人
10. account\_uid\_type           from;         //转出的账户
11. asset                      amount;       //资产数量
12. uint8\_t                    benefit\_type; //类型
13. optional<account\_uid\_type> to;           //转到的账户
14. optional<time\_point\_sec>   time;         //时间
16. extensions\_type            extensions;   //扩展参数
17. };

新增benefit\_collect\_operation，用于账户的受益人领取收益，收益的类型为见证人工资或锁仓产生的积分。

# 5. Database\_api 修改整理

## 5.1 新增list\_pledge\_mining\_by\_witness

1. vector<pledge\_mining\_object> list\_pledge\_mining\_by\_witness(account\_uid\_type   witness,             //抵押见证人
2. account\_uid\_type   lower\_bound\_account, //起始抵押账户id
3. uint32\_t limit)；                       //展示数目限制

更新说明：获取抵押给该见证人的所有抵押详情

## 5.2 新增list\_pledge\_mining\_by\_account

1. vector<pledge\_mining\_object> list\_pledge\_mining\_by\_account(account\_uid\_type   account,             //抵押账户
2. account\_uid\_type   lower\_bound\_witness, //起始抵押见证人
3. uint32\_t limit)；                       //展示数目限制

更新说明：获取该账户抵押给所有见证人的抵押详情

## 5.3 新增get\_limit\_orders

1. vector<limit\_order\_object> get\_limit\_orders(std::string a, //资产a的符号
2. std::string b, //资产b的符号
3. uint32\_t limit //展示数目限制
4. )**const**;

更新说明：查询资产交易限价订单

## 5.4 新增get\_account\_limit\_orders

1. vector<limit\_order\_object> get\_account\_limit\_orders(**const** string& account\_name\_or\_id, //账户的id或名字
2. **const** string &base,               //资产a的符号
3. **const** string  e,               //资产b的符号
4. uint32\_t limit = 101,             //展示数目限制
5. optional<limit\_order\_id\_type> ostart\_id = optional<limit\_order\_id\_type>(), //要查询起始的limit\_order的id
6. optional<price> ostart\_price = optional<price>());                         //要查询的报价

更新说明：通过账户查询资产交易限价订单

## 5.5 新增get\_ticker

1. market\_ticker get\_ticker(**const** string& base, //资产a的符号
2. **const** string& quote //资产b的符号
3. )**const**;

更新说明：查询资产交易行情

## 5.6 新增get\_24\_volume

1. market\_volume get\_24\_volume(**const** string& base, //资产a的符号
2. **const** string& quote //资产b的符号
3. )**const**;

更新说明：查询资产交易容量

## 5.7 新增get\_order\_book

1. order\_book get\_order\_book(**const** string& base,  //资产a的符号
2. **const** string& quote, //资产b的符号
3. unsigned limit = 50  //展示数目限制
4. )**const**;

更新说明：查询资产交易的交易单

## 5.8 新增get\_top\_markets

1. vector<market\_ticker> get\_top\_markets(uint32\_t limit //展示数目限制
2. )**const**;

更新说明：查询前列资产交易市场行情

## 5.9 新增get\_trade\_history

1. vector<market\_trade> get\_trade\_history(**const** string& base,       //资产a的符号
2. **const** string& quote,      //资产b的符号
3. fc::time\_point\_sec start, //截至最新时间
4. fc::time\_point\_sec stop,  //查询最早时间
5. unsigned limit = 100      //展示数目限制
6. )**const**;

更新说明：查询资产交易历史

## 5.10 新增get\_trade\_history\_by\_sequence

1. vector<market\_trade> get\_trade\_history\_by\_sequence(**const** string& base,      //资产a的符号
2. **const** string& quote,     //资产b的符号
3. int64\_t start,           //截至最新的序列号
4. fc::time\_point\_sec stop, //查询最早的时间
5. unsigned limit = 100     //展示数目限制
6. )**const**;

更新说明：通过序列号查询资产交易的历史

## 5.11 新增subscribe\_to\_market

1. **void** subscribe\_to\_market(std::function<**void**(**const** variant&)> callback, //回调接口
2. **const** std::string& a,                         //资产a的符号
3. **const** std::string& b                          //资产b的符号
4. );

更新说明：订阅资产交易

## 5.12 新增unsubscribe\_from\_market

1. **void** unsubscribe\_from\_market(**const** std::string& a, //资产a的符号
2. **const** std::string& b  //资产b的符号
3. );

更新说明：取消资产交易订阅

## 5.13 新增compute\_coin\_seconds\_earned

1. std::pair<fc::uint128\_t, share\_type> compute\_coin\_seconds\_earned(**const** account\_uid\_type uid,  //账户的id
2. **const** uint64\_t window,       //时间窗口
3. **const** fc::time\_point\_sec now //现在的时间
4. )**const**;

更新说明：新增compute\_coin\_seconds\_earned，用于计算账户赚取的积分和时间的乘积。因hard\_fork05之后账户的积分由所有有效资产和时间的乘积产生，改为了账户抵押资产产生积分。因此，新增积分计算接口供wallet调用。

## 5.14 新增get\_account\_core\_asset\_pledge

1. vector<pledge\_balance\_object> get\_account\_core\_asset\_pledge(account\_uid\_type account\_uid  //账户的id
2. )**const**;

更新说明：新增get\_account\_core\_asset\_pledge，用于查询账户核心资产抵押的详情。内容包括见证人抵押，理事会抵押，平台抵押，积分抵押，以及挖矿抵押。

## 5.15 新增get\_account\_all\_limit\_orders

1. vector<limit\_order\_object> get\_account\_all\_limit\_orders(**const** string& name\_or\_id, //账户名
2. uint32\_t limit = 101,   //显示数目限制
3. optional<limit\_order\_id\_type> ostart\_id); //起始id

更新说明：新增get\_account\_all\_limit\_orders，用于查询账户所有交易对的限价订单详情。

# 6. History\_api 修改整理

## 6.1 新增get\_market\_history

1. vector<bucket\_object> get\_market\_history(std::string a,                 //资产a的符号
2. std::string b,                 //资产b的符号
3. uint32\_t bucket\_seconds,       //时间间隔
4. fc::time\_point\_sec start,      //截至最新时间
5. fc::time\_point\_sec end)**const**;  //查询最早时间

更新说明：查询资产交易市场历史

# 7. Wallet修改整理

## 7.1 新增create\_limit\_order

1. signed\_transaction create\_limit\_order(string           seller,                   //售出资产者
2. asset\_aid\_type   sell\_asset\_symbol,        //售出资产符号
3. share\_type       sell\_amount,              //售出数量
4. asset\_aid\_type   min\_receive\_asset\_symbol, //接收资产符号
5. share\_type       min\_receive\_amount,       //最少接收数量
6. uint32\_t         expiration,               //订单过期时间
7. **bool**             fill\_or\_kill,             //true：若没匹配上则取消；false：没匹配上暂存在链上
8. **bool**             broadcast = **false**);       //是否广播

更新说明：新增创建限价订单，用于资产交易

示例：*create\_limit\_order test YOYO 10000 ABC 50000 1579862861 false true*

## 7.2 新增cancel\_limit\_order

1. signed\_transaction cancel\_limit\_order(string               seller,             //限价订单发起者
2. limit\_order\_id\_type  order\_id,           //限价订单的id
3. **bool**                 broadcast = **false**); //是否广播

更新说明：新增取消限价订单，用于取消待处理的限价订单

示例：*cancel\_limit\_order test 1.12.0 true*

## 7.3 新增collect\_market\_fee

1. signed\_transaction collect\_market\_fee(string               account,           //取回市场费用的账户
2. string               asset\_symbol,      //取回资产的符号
3. string               amount,            //取回市场费用的数量
4. **bool**                 csaf\_fee = **true**,   //使用积分付手续费。true，积分付手续费；false，余额付手续费
5. **bool**                 broadcast = **false**  //是否广播
6. );

更新说明：新增取回市场交易手续费产生的收益

示例：*collect\_market\_fee test YOYO 10000 false true*

## 7.4 新增asset\_claim\_fees

1. signed\_transaction asset\_claim\_fees(string               issuer,            //发起人
2. string               asset\_symbol,      //资产的符号
3. string               amount\_to\_claim,   //收回的资产交易市场费用
4. **bool**                 csaf\_fee = **true**,   //使用积分付手续费。true，积分付手续费；false，余额付手续费
5. **bool**                 broadcast = **false**  //是否广播
6. );

更新说明：新增资产创建人取回该资产市场交易产生的手续费

示例：*asset\_claim\_fees test YOYO 10000 false true*

## 7.5 新增update\_lock\_balance

1. signed\_transaction update\_lock\_balance(string lock\_balance\_account, //抵押发起的账户id或名字
2. string lock\_balance\_amount,  //抵押的额度
3. **bool** csaf\_fee = **true**,        //是否使用积分支付手续费
4. **bool** broadcast = **false**);     //是否广播

更新说明：新增创建，更新或取消锁仓产出积分

示例： *update\_lock\_balance test\_account1 1000 true true*

## 7.6 新增update\_mining\_pledge

1. signed\_transaction update\_mining\_pledge(string pledge\_account,     //抵押账户
2. string witness,            //抵押见证人
3. uint64\_t new\_pledge,       //抵押量
4. **bool** csaf\_fee,             //是否使用积分支付手续费
5. **bool** broadcast);           //是否广播

更新说明：新增挖矿抵押，创建，更新或取消挖矿抵押

示例：update\_mining\_pledge test witness1 10000000000 true true

## 7.7 新增collect\_pledge\_mining\_bonus

1. signed\_transaction collect\_pledge\_mining\_bonus(string collect\_account,  //收集分红账户
2. uint64\_t bonus\_amount,   //收集分红金额
3. **bool** csaf\_fee,           //是否使用积分支付手续费
4. **bool** broadcast)；        //是否广播

更新说明：收集抵押挖矿分红

示例：collect\_pledge\_mining\_bonus test 10000000000 true true

## 7.8 新增list\_pledge\_mining\_by\_witness

1. vector<pledge\_mining\_object> list\_pledge\_mining\_by\_witness(string   witness,                       //抵押见证人
2. account\_uid\_type   lower\_bound\_account, //起始抵押账户id
3. uint32\_t limit)；                       //展示数目限制

更新说明：获取抵押给该见证人的所有抵押详情

示例：list\_pledge\_mining\_by\_witness witness1 0 10

## 7.9 新增list\_pledge\_mining\_by\_account

1. vector<pledge\_mining\_object> list\_pledge\_mining\_by\_account(string   account,                       //抵押账户
2. account\_uid\_type   lower\_bound\_witness, //起始抵押见证人
3. uint32\_t limit)；                       //展示数目限制

更新说明：获取该账户抵押给所有见证人的抵押详情

示例：list\_pledge\_mining\_by\_account test1 0 10

## 7.10 新增collect\_score\_bonus

1. signed\_transaction collect\_score\_bonus(string collect\_account,  //收集分红账户
2. uint64\_t bonus\_amount,   //收集分红金额
3. **bool** csaf\_fee,           //是否使用积分支付手续费
4. **bool** broadcast)；        //是否广播

更新说明：注册商、引荐人收集点赞分红

示例：collect\_score\_bonus test 10000000000 true true

## 7.11 新增get\_limit\_orders

1. vector<limit\_order\_object> get\_limit\_orders(std::string a, //资产a的符号
2. std::string b, //资产b的符号
3. uint32\_t limit //展示数目限制
4. )**const**;

更新说明：获取资产交易限价订单

示例：get\_limit\_orders ABC CBA 50

## 7.12 新增get\_account\_limit\_orders

1. vector<limit\_order\_object> get\_account\_limit\_orders(**const** string& account\_name\_or\_id, //账户的id或名字
2. **const** string &base,               //资产a的符号
3. **const** string e,               //资产b的符号
4. uint32\_t limit = 101,             //展示数目限制
5. optional<limit\_order\_id\_type> ostart\_id = optional<limit\_order\_id\_type>(), //要查询起始的limit\_order的id
6. optional<price> ostart\_price = optional<price>());                         //要查询的报价

更新说明：获取账户的资产交易限价订单

示例：get\_account\_limit\_orders test2 ABC CBA 50 null null

## 7.13 新增get\_ticker

1. market\_ticker get\_ticker(**const** string& base, //资产a的符号
2. **const** string& quote //资产b的符号
3. )**const**;

更新说明：获取资产交易行情

示例：get\_ticker ABC CBA

## 7.14 新增get\_24\_volume

1. market\_volume get\_24\_volume(**const** string& base, //资产a的符号
2. **const** string& quote //资产b的符号
3. )**const**;

更新说明：获取市场容量

示例： get\_24\_volume ABC CBA

## 7.15 新增get\_order\_book

1. order\_book get\_order\_book(**const** string& base,  //资产a的符号
2. **const** string& quote, //资产b的符号
3. unsigned limit = 50  //展示数目限制
4. )**const**;

更新说明：获取资产市场交易单

示例：get\_order\_book ABC CBA 50

## 7.16 新增get\_top\_markets

1. vector<market\_ticker> get\_top\_markets(uint32\_t limit //展示数目限制
2. )**const**;

更新说明：获取市场行情

示例：get\_top\_markets 50

## 7.17 新增get\_trade\_history

1. vector<market\_trade> get\_trade\_history(**const** string& base,       //资产a的符号
2. **const** string& quote,      //资产b的符号
3. uint32\_t start,  //截至最新时间
4. uint32\_t stop,   //查询最早时间
5. unsigned limit = 100      //展示数目限制
6. )**const**;

更新说明：获取资产交易历史

示例：get\_trade\_history ABC CBA 1585020800 1555193600 100

## 7.18 新增get\_trade\_history\_by\_sequence

1. vector<market\_trade> get\_trade\_history\_by\_sequence(**const** string& base,      //资产a的符号
2. **const** string& quote,     //资产b的符号
3. int64\_t start,           //截至最新的序列号
4. uint32\_t stop,  //查询最早的时间
5. unsigned limit = 100     //展示数目限制
6. )**const**;

更新说明：依据序列号获取资产交易历史

示例：get\_trade\_history\_by\_sequence ABC CBA 3 1535193600 100

## 7.19 新增get\_market\_history

1. vector<bucket\_object> get\_market\_history(std::string a,                 //资产a的符号
2. std::string b,                 //资产b的符号
3. uint32\_t bucket\_seconds,       //时间间隔
4. uint32\_t start,       //截至最新时间
5. uint32\_t end)**const**;   //查询最早时间

更新说明：获取市场历史

示例：get\_market\_history ABC CBA 60 1534934400 1585193600

## 7.20 新增get\_account\_core\_asset\_pledge

1. vector<pledge\_balance\_object> get\_account\_core\_asset\_pledge(account\_uid\_type account\_uid  //账户的id
2. )**const**;

更新说明：新增get\_account\_core\_asset\_pledge，用于查询账户核心资产抵押的详情。内容包括见证人抵押，理事会抵押，平台抵押，积分抵押，以及挖矿抵押。

示例：get\_account\_core\_asset\_pledge test\_account

## 7.21 新增get\_account\_all\_limit\_orders

1. vector<limit\_order\_object> get\_account\_all\_limit\_orders(**const** string& name\_or\_id, //账户名
2. uint32\_t limit = 101,  //显示数目限制
3. optional<limit\_order\_id\_type> ostart\_id); //起始id

更新说明：新增get\_account\_all\_limit\_orders，用于查询账户的所有交易对的限价订单详情。

示例：get\_account\_all\_limit\_orders test\_account 100 1.12.0

## 7.22 新增assign\_beneficiary

1. signed\_transaction assign\_beneficiary(string   account,            //账户名
2. string   new\_beneficiary,    //新的受益人
3. **bool**     csaf\_fee = **true**,    //是否使用积分付手续费
4. **bool**     broadcast = **false**); //是否广播

更新说明：新增assign\_beneficiary，用于账户指定受益人。

示例：assign\_beneficiary test1 test2 true true

## 7.23 新增collect\_benefit

1. signed\_transaction collect\_benefit(string              issuer,             //受益人
2. string              from,               //转出账户
3. uint8\_t             benefit\_type,       //收益类型
4. string              amount,             //资产数量
5. string              asset\_symbol,       //资产符号
6. optional<string>    to,                 //转到的账户
7. optional<uint32\_t>  time,               //时间
8. **bool**                csaf\_fee =  **true**,   //是否使用积分付手续费
9. **bool**                broadcast = **false**); //是否广播

更新说明：新增collect\_benefit，用于账户的受益人领取收益。

示例：collect\_benefit test1 test2 1 1000 YOYO null null true true

# 8. 启动命令行参数修改

将enable\_check\_invariants改为 check\_invariants\_interval的类型为了uint32\_t

示例：--check\_invariants\_interval = 9 表示每隔9个块，做一次全局供应量检查。

# 9. 理事会提案类型增加

1. **struct** committee\_withdraw\_platform\_pledge\_item\_type
2. {
3. account\_uid\_type    platform\_account;    //受处罚平台账户
4. account\_uid\_type    receiver;            //处罚金接受账户
5. share\_type          withdraw\_amount;     //处罚金额
6. extensions\_type     extensions;          //扩展
8. **void** validate()**const**;
9. };
10. **typedef** fc::static\_variant <
11. committee\_update\_account\_priviledge\_item\_type,
12. committee\_update\_fee\_schedule\_item\_type,
13. committee\_update\_global\_parameter\_item\_type,
14. committee\_update\_global\_content\_parameter\_item\_type,
15. committee\_withdraw\_platform\_pledge\_item\_type         //撤回平台抵押理事会提案类型
16. > committee\_proposal\_item\_type;

新增撤回平台抵押理事会提案类型，如果平台存在作恶，将用户授权给该平台的零钱转走，理事会可以发起提案，撤走平台的抵押，用于维护良好的内容收益循环环境。如果理事会设置撤回金额大于平台的全部抵押与待释放抵押之后，则撤回金额为平台的全部抵押与待释放抵押之和；撤回抵押优先从待释放的抵押之中扣除；如果撤走部分抵押之后，并且剩余抵押小于平台最低抵押，那么删除平台对象，但是平台授权对象仍然存在；