PBSA

Profit-Sharing on the Peerplays Blockchain: A Technical Overview

This overview document covers the blockchain-level functions of the Peerplays Profit-Sharing module.

WALLET COMMAND

There is one wallet command the user can use to control a dividend-paying asset:

update_dividend_asset asset-symbol new-options broadcast

This function creates a distribution account on the blockchain, which is where all the funds to be distributed to the dividend-paying asset will accumulate.

If the dividend-paying asset is a UIA (User-Issued Asset) then the distribution account will be controlled by the individual or multi-sig user(s) that issued the UIA.

If the dividend-paying asset is an FBA (fee-backed asset), then the distribution account will be controlled by the FBA committee.

If the dividend-paying asset is the core token of the blockchain, then the distribution account will be controlled by the general blockchain committee.

Note: Since Peerplays is implementing this feature at the core token level, many of the network fees will be routed directly into the distribution account.

If the asset (UIA, FBA, or core) is currently a dividend-paying asset, the above wallet command will update its options. If the asset is not, it will convert the asset into a dividend-paying asset with the given options.

The new account will be named something like [asset-name]-dividend-distribution. (A suffix should be chosen that is currently unused, and will thus be reserved for future use with all dividend-paying assets on the chain).

OPTIONS

These are user-settable & sent on the blockchain:

fc::optional<fc::time_point_sec> next_payout_time – This sets the time when the next payout should occur. The payouts will happen on the *blockchain maintenance interval* – at or after this time. If this is set to null, there will be no payouts.

fc::optional<uint32_t> payout_interval – If payouts are to happen on a fixed schedule, this specifies the interval between payouts in seconds. After each payout, the next payout time will be incremented by this amount. If payout_interval is not set, the next payout (if any) will be the last until the options are updated again.

uint64_t minimum_fee_percentage – Each dividend distribution incurs a fee that is based on the number of accounts that hold the dividend asset, rather than a percentage of the amount paid out. This parameter prevents assets from being distributed unless the fee is less than the percentage set here, to prevent a slow trickle of deposits to the account from being completely consumed in fees. In other words, if you set this parameter to 10% and the fees work out to 100 CORE to share out, balances in the dividend distribution accounts will not be shared out if the balance is less than 10000 CORE.

Impact of the Peerplays Maintenance Interval
Normally, pending dividend payments are calculated each maintenance interval in which there are balances in the dividend distribution account. At present, this happens to be once per hour on the BitShares blockchain. If this is too often (too expensive in fees or to computationally-intensive for the blockchain) this can be increased to once every [n] hours. For example, if you set this to one day, distributions will take place on even multiples of one day, allowing deposits to the distribution account to accumulate for 23 maintenance intervals and then computing the pending payouts on the 24th.
Payouts will always occur at the next payout time whether or not it falls on a multiple of the distribution interval, and the timer on the distribution interval is reset at payout time. So if you have the distribution interval at three days and the payout interval at one week, payouts will occur at days 3, 6, 7, 10, 13, 14..etc.

ADDITIONAL DATA STORED IN MEMORY PER ASSET

(This is maintained by each node for reporting to the user, in asset_dividend_data_object)

fc::optional<time_point_sec> last_payout_time – The time payouts on this asset were last proceed

fc::optional<time_point_sec> last_distribution_time – The time pending payouts on this asset were last processed

account_id_type dividend_collecting_account – The account which collects pending payouts

Other data stored in asset_dividend_data_object for facilitating calculations:

Asset names – are currently defined as 3 – 16 characters, starting and ending with a letter, possibly containing a dot. if there’s a dot, the part before the dot will have to conform to the same rules (at least 3 letters), and defined in is_valid_symbol() called by asset_create_operation::validate(), and more completely in asset_create_evaluator::do_evaluate()

Account names – are currently defined in is_valid_name() called by account_create_operation::validate() and in account_create_evaluator::do_evaluate()

INDEXES

Scheduled Payouts Index records all the payouts that are scheduled to take place (as of last maintenance interval):

dividend_asset I payout asset I user_id I balance

Already Scheduled Balances Index keeps track of the balance of each asset in each dividend-paying account as of the last time we computed scheduled payments (last maintenance interval):

dividend_asset I payout asset I balance

DIVIDEND-DISTRIBUTION ACCOUNT

The dividend distribution account is a regular account created by the system at the time the dividend-paying asset is created. The account has no owner, so anyone user can transfer to this account but no user can transfer from this account. Each asset has only one dividend distribution account. Any balances in this account will be shared out to dividend- asset holders on the payout date. The account can hold any asset (the core asset, user-issued assets, even the associated dividend-paying asset.)

EX-DIVIDEND DISTRIBUTIONS

Ex-dividend distributions are computed at each maintenance interval when there has been a transfer to the dividend distribution account.

FEES

Ex-dividend calculations impose a computational load on the BitShares network (slowing down re-indexing), so the fees are designed to compensate the network for the work. There are two fee parameters associated with dividend-paying assets:

dividend_distribution_base_fee

dividend_distribution_fee_per_holder

A fee is charged for each ex-dividend distribution in each asset distributed. The fee is taken from the amount being distributed. Fees are always paid in the core asset. Assets other than the core asset can only be distributed by using their core exchange rate and fee pool to exchange the asset for the core asset.

For example, suppose all fee parameters were set to 1 CORE and we are distributing the balances of the distribution account for a dividend- paying asset which is held in equal amounts by 100 users. If there is 5101 CORE in the distribution account when the maintenance interval arrives, the system will calculate the fee of 101 CORE (1 base fee + 100 * 1 per-holder fee). This fee is paid to the system, then the remaining 5000 CORE is distributed among the holders; in this case, 50 CORE to each holder.

RESTRICTED ASSETS

When a processing dividend distribution of a restricted asset, the normal pending dividend distribution calculations will be made assuming all users are able to receive the asset. When the dividend payout time arrives, any pending dividend distributions to prohibited accounts are voided, and their pending amounts are redistributed to the remaining accounts.

With this setup, a dividend-paying asset holder is not penalized for any time their account is frozen – they only miss out if their account is frozen at the time the asset pays out.

OVERRIDE TRANSFERS

For any UIAs that have the issuer override flag set, the owner of the asset can transfer that asset from any account, even if they don't own the account. This makes it possible for the asset issuer to take back funds they have been sent to the distribution account and for which that the blockchain has already computed pending payouts. In these cases, all pending payments will be reduced in proportion to their share of the total pending payments, instead of by their share of the dividend asset.

This is expected to be a rare event, and any asset that abuses its override privileges in this way will likely be of little value.

At each maintenance interval, we loop over each dividend-paying asset.

For each asset:
Examine the balance of the associated account, and compute the difference between the current balance in each asset compared to the balance at the last maintenance interval. These delta-balances are what we need to schedule for payout.

For each delta-balance:
If the delta-balance is positive, compute the fee required to distribute that asset, and compare it to the amount that will be distributed. If the fee is a sufficiently small percentage of the distributed amount, the distribution is done.

The distribution:
Walk through each of the owners of the dividend-paying asset. Calculate their share of each payout asset as (their_balance_of_dividend_paying_asset / total_dividend_paying_asset_issued) * delta_balance. Then increase their scheduled payout of that asset by the calculated amount.

If the delta-balance is negative:
Walk through all scheduled payouts of that asset. Calculate the share each user must return as (their_current_scheduled_payout / *total_scheduled_payouts) * -delta_balance. Then decrease their scheduled payout of that asset by the calculated amount.

If the next_payout time has passed:
Walk through all scheduled payouts. Increase the dividend-asset holder's balance by their scheduled amount and decrease the dividend payout account's balance by their scheduled amount. Decrease the scheduled payout for that account by the amount they were paid. Generate a virtual op showing a dividend payout of that amount Set last_payout_time to now.

If payout_interval is set, bump next_payout_time up by the interval, otherwise clear next_payout_time.