

```

1 // SPDX-License-Identifier: AGPL-3.0-or-later
2
3 pragma solidity 0.7.5;
4
5 library LowGasSafeMath {
6     /// @notice Returns x + y, reverts if sum overflows uint256
7     /// @param x The augend
8     /// @param y The addend
9     /// @return z The sum of x and y
10    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
11        require((z = x + y) >= x);
12    }
13
14    function add32(uint32 x, uint32 y) internal pure returns (uint32 z) {
15        require((z = x + y) >= x);
16    }
17
18    /// @notice Returns x - y, reverts if underflows
19    /// @param x The minuend
20    /// @param y The subtrahend
21    /// @return z The difference of x and y
22    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
23        require((z = x - y) <= x);
24    }
25
26    function sub32(uint32 x, uint32 y) internal pure returns (uint32 z) {
27        require((z = x - y) <= x);
28    }
29
30    /// @notice Returns x * y, reverts if overflows
31    /// @param x The multiplicand
32    /// @param y The multiplier
33    /// @return z The product of x and y
34    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
35        require(x == 0 || (z = x * y) / x == y);
36    }
37
38    /// @notice Returns x + y, reverts if overflows or underflows
39    /// @param x The augend
40    /// @param y The addend
41    /// @return z The sum of x and y
42    function add(int256 x, int256 y) internal pure returns (int256 z) {
43        require((z = x + y) >= x == (y >= 0));
44    }
45
46    /// @notice Returns x - y, reverts if overflows or underflows
47    /// @param x The minuend
48    /// @param y The subtrahend
49    /// @return z The difference of x and y
50    function sub(int256 x, int256 y) internal pure returns (int256 z) {
51        require((z = x - y) <= x == (y >= 0));
52    }
53

```

```

1 // SPDX-License-Identifier: AGPL-3.0-or-later
2
3 pragma solidity 0.7.5;
4
5 library LowGasSafeMath {
6     /// @notice Returns x + y, reverts if sum overflows uint256
7     /// @param x The augend
8     /// @param y The addend
9     /// @return z The sum of x and y
10    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
11        require((z = x + y) >= x);
12    }
13
14    function add32(uint32 x, uint32 y) internal pure returns (uint32 z) {
15        require((z = x + y) >= x);
16    }
17
18    /// @notice Returns x - y, reverts if underflows
19    /// @param x The minuend
20    /// @param y The subtrahend
21    /// @return z The difference of x and y
22    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
23        require((z = x - y) <= x);
24    }
25
26    function sub32(uint32 x, uint32 y) internal pure returns (uint32 z) {
27        require((z = x - y) <= x);
28    }
29
30    /// @notice Returns x * y, reverts if overflows
31    /// @param x The multiplicand
32    /// @param y The multiplier
33    /// @return z The product of x and y
34    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
35        require(x == 0 || (z = x * y) / x == y);
36    }
37
38    /// @notice Returns x + y, reverts if overflows or underflows
39    /// @param x The augend
40    /// @param y The addend
41    /// @return z The sum of x and y
42    function add(int256 x, int256 y) internal pure returns (int256 z) {
43        require((z = x + y) >= x == (y >= 0));
44    }
45
46    /// @notice Returns x - y, reverts if overflows or underflows
47    /// @param x The minuend
48    /// @param y The subtrahend
49    /// @return z The difference of x and y
50    function sub(int256 x, int256 y) internal pure returns (int256 z) {
51        require((z = x - y) <= x == (y >= 0));
52    }
53

```

```

54     function div(uint256 x, uint256 y) internal pur
e returns(uint256 z){
55         require(y > 0);
56         z=x/y;
57     }
58 }
59
60 interface IERC20 {
61
62     function totalSupply() external view returns (u
int256);
63
64     function balanceOf(address account) external vi
ew returns (uint256);
65
66     function transfer(address recipient, uint256 am
ount) external returns (bool);
67
68     function allowance(address owner, address spend
er) external view returns (uint256);
69
70     function approve(address spender, uint256 amoun
t) external returns (bool);
71
72     function transferFrom(address sender, address r
ecipient, uint256 amount) external returns (bool);
73
74     event Transfer(address indexed from, address in
dexed to, uint256 value);
75
76     event Approval(address indexed owner, address i
ndexed spender, uint256 value);
77 }
78
79 library Address {
80
81     function isContract(address account) internal v
iew returns (bool) {
82         // This method relies in extcodesize, which
returns 0 for contracts in
83         // construction, since the code is only sto
red at the end of the
84         // constructor execution.
85
86         uint256 size;
87         // solhint-disable-next-line no-inline-asse
mbly
88         assembly { size := extcodesize(account) }
89         return size > 0;
90     }
91
92     function sendValue(address payable recipient, u
int256 amount) internal {
93         require(address(this).balance >= amount, "A
ddress: insufficient balance");
94
95         // solhint-disable-next-line avoid-low-leve
l-calls, avoid-call-value
96         (bool success, ) = recipient.call{ value: a
mount }("");
97         require(success, "Address: unable to send v
alue, recipient may have reverted");
98     }
99
100    function functionCall(address target, bytes mem
ory data) internal returns (bytes memory) {
101        return functionCall(target, data, "Address: l
ow-level call failed");
102    }
103
104    function functionCall(

```

```

54     function div(uint256 x, uint256 y) internal pur
e returns(uint256 z){
55         require(y > 0);
56         z=x/y;
57     }
58 }
59
60 interface IERC20 {
61
62     function totalSupply() external view returns (u
int256);
63
64     function balanceOf(address account) external vi
ew returns (uint256);
65
66     function transfer(address recipient, uint256 am
ount) external returns (bool);
67
68     function allowance(address owner, address spend
er) external view returns (uint256);
69
70     function approve(address spender, uint256 amoun
t) external returns (bool);
71
72     function transferFrom(address sender, address r
ecipient, uint256 amount) external returns (bool);
73
74     event Transfer(address indexed from, address in
dexed to, uint256 value);
75
76     event Approval(address indexed owner, address i
ndexed spender, uint256 value);
77 }
78
79 library Address {
80
81     function isContract(address account) internal v
iew returns (bool) {
82         // This method relies in extcodesize, which
returns 0 for contracts in
83         // construction, since the code is only sto
red at the end of the
84         // constructor execution.
85
86         uint256 size;
87         // solhint-disable-next-line no-inline-asse
mbly
88         assembly { size := extcodesize(account) }
89         return size > 0;
90     }
91
92     function sendValue(address payable recipient, u
int256 amount) internal {
93         require(address(this).balance >= amount, "A
ddress: insufficient balance");
94
95         // solhint-disable-next-line avoid-low-leve
l-calls, avoid-call-value
96         (bool success, ) = recipient.call{ value: a
mount }("");
97         require(success, "Address: unable to send v
alue, recipient may have reverted");
98     }
99
100    function functionCall(address target, bytes mem
ory data) internal returns (bytes memory) {
101        return functionCall(target, data, "Address: l
ow-level call failed");
102    }
103
104    function functionCall(

```

```

105     address target,
106     bytes memory data,
107     string memory errorMessage
108 ) internal returns (bytes memory) {
109     return _functionCallWithValue(target, data,
110     0, errorMessage);
111 }
112 function functionCallWithValue(address target,
113     bytes memory data, uint256 value) internal returns
114 (bytes memory) {
115     return functionCallWithValue(target, data,
116     value, "Address: low-level call with value faile
117 d");
118 }
119 function functionCallWithValue(
120     address target,
121     bytes memory data,
122     uint256 value,
123     string memory errorMessage
124 ) internal returns (bytes memory) {
125     require(address(this).balance >= value, "Ad
126 dress: insufficient balance for call");
127     require(isContract(target), "Address: call
128 to non-contract");
129
130     // solhint-disable-next-line avoid-low-leve
131 l-calls
132     (bool success, bytes memory returndata) = t
133 arget.call{ value: value }(data);
134     return _verifyCallResult(success, returndat
135 a, errorMessage);
136 }
137
138 function _functionCallWithValue(
139     address target,
140     bytes memory data,
141     uint256 weiValue,
142     string memory errorMessage
143 ) private returns (bytes memory) {
144     require(isContract(target), "Address: call
145 to non-contract");
146
147     // solhint-disable-next-line avoid-low-leve
148 l-calls
149     (bool success, bytes memory returndata) = t
150 arget.call{ value: weiValue }(data);
151     if (success) {
152         return returndata;
153     } else {
154         // Look for revert reason and bubble it
155 up if present
156         if (returndata.length > 0) {
157             // The easiest way to bubble the re
158 vert reason is using memory via assembly
159
160             // solhint-disable-next-line no-inl
161 ine-assembly
162             assembly {
163                 let returndata_size := mload(re
164 turndata)
165                 revert(add(32, returndata), ret
166 urndata_size)
167             }
168         } else {
169             revert(errorMessage);
170         }
171     }
172 }

```

```

105     address target,
106     bytes memory data,
107     string memory errorMessage
108 ) internal returns (bytes memory) {
109     return _functionCallWithValue(target, data,
110     0, errorMessage);
111 }
112 function functionCallWithValue(address target,
113     bytes memory data, uint256 value) internal returns
114 (bytes memory) {
115     return functionCallWithValue(target, data,
116     value, "Address: low-level call with value faile
117 d");
118 }
119 function functionCallWithValue(
120     address target,
121     bytes memory data,
122     uint256 value,
123     string memory errorMessage
124 ) internal returns (bytes memory) {
125     require(address(this).balance >= value, "Ad
126 dress: insufficient balance for call");
127     require(isContract(target), "Address: call
128 to non-contract");
129
130     // solhint-disable-next-line avoid-low-leve
131 l-calls
132     (bool success, bytes memory returndata) = t
133 arget.call{ value: value }(data);
134     return _verifyCallResult(success, returndat
135 a, errorMessage);
136 }
137
138 function _functionCallWithValue(
139     address target,
140     bytes memory data,
141     uint256 weiValue,
142     string memory errorMessage
143 ) private returns (bytes memory) {
144     require(isContract(target), "Address: call
145 to non-contract");
146
147     // solhint-disable-next-line avoid-low-leve
148 l-calls
149     (bool success, bytes memory returndata) = t
150 arget.call{ value: weiValue }(data);
151     if (success) {
152         return returndata;
153     } else {
154         // Look for revert reason and bubble it
155 up if present
156         if (returndata.length > 0) {
157             // The easiest way to bubble the re
158 vert reason is using memory via assembly
159
160             // solhint-disable-next-line no-inl
161 ine-assembly
162             assembly {
163                 let returndata_size := mload(re
164 turndata)
165                 revert(add(32, returndata), ret
166 urndata_size)
167             }
168         } else {
169             revert(errorMessage);
170         }
171     }
172 }

```

```

156     }
157
158     function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
159         return functionStaticCall(target, data, "Address: low-level static call failed");
160     }
161
162     function functionStaticCall(
163         address target,
164         bytes memory data,
165         string memory errorMessage
166     ) internal view returns (bytes memory) {
167         require(isContract(target), "Address: static call to non-contract");
168
169         // solhint-disable-next-line avoid-low-level-calls
170         (bool success, bytes memory returndata) = target.staticcall(data);
171         return _verifyCallResult(success, returndata, errorMessage);
172     }
173
174     function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
175         return functionDelegateCall(target, data, "Address: low-level delegate call failed");
176     }
177
178     function functionDelegateCall(
179         address target,
180         bytes memory data,
181         string memory errorMessage
182     ) internal returns (bytes memory) {
183         require(isContract(target), "Address: delegate call to non-contract");
184         (bool success, bytes memory returndata) = target.delegatecall(data);
185         return _verifyCallResult(success, returndata, errorMessage);
186     }
187
188     function _verifyCallResult(
189         bool success,
190         bytes memory returndata,
191         string memory errorMessage
192     ) private pure returns (bytes memory) {
193         if (success) {
194             return returndata;
195         } else {
196             if (returndata.length > 0) {
197                 assembly {
198                     let returndata_size := mload(returndata)
199                     revert(add(32, returndata), returndata_size)
200                 }
201             } else {
202                 revert(errorMessage);
203             }
204         }
205     }
206
207     function addressToString(address _address) internal pure returns (string memory) {
208         bytes32 _bytes = bytes32(uint256(_address));

```

```

156     }
157
158     function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
159         return functionStaticCall(target, data, "Address: low-level static call failed");
160     }
161
162     function functionStaticCall(
163         address target,
164         bytes memory data,
165         string memory errorMessage
166     ) internal view returns (bytes memory) {
167         require(isContract(target), "Address: static call to non-contract");
168
169         // solhint-disable-next-line avoid-low-level-calls
170         (bool success, bytes memory returndata) = target.staticcall(data);
171         return _verifyCallResult(success, returndata, errorMessage);
172     }
173
174     function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
175         return functionDelegateCall(target, data, "Address: low-level delegate call failed");
176     }
177
178     function functionDelegateCall(
179         address target,
180         bytes memory data,
181         string memory errorMessage
182     ) internal returns (bytes memory) {
183         require(isContract(target), "Address: delegate call to non-contract");
184         (bool success, bytes memory returndata) = target.delegatecall(data);
185         return _verifyCallResult(success, returndata, errorMessage);
186     }
187
188     function _verifyCallResult(
189         bool success,
190         bytes memory returndata,
191         string memory errorMessage
192     ) private pure returns (bytes memory) {
193         if (success) {
194             return returndata;
195         } else {
196             if (returndata.length > 0) {
197                 assembly {
198                     let returndata_size := mload(returndata)
199                     revert(add(32, returndata), returndata_size)
200                 }
201             } else {
202                 revert(errorMessage);
203             }
204         }
205     }
206
207     function addressToString(address _address) internal pure returns (string memory) {
208         bytes32 _bytes = bytes32(uint256(_address));

```

```

209     bytes memory HEX = "0123456789abcdef";
210     bytes memory _addr = new bytes(42);
211
212     _addr[0] = '0';
213     _addr[1] = 'x';
214
215     for(uint256 i = 0; i < 20; i++) {
216         _addr[2+i*2] = HEX[uint8(_bytes[i + 12]
>> 4)];
217         _addr[3+i*2] = HEX[uint8(_bytes[i + 12]
& 0x0f)];
218     }
219
220     return string(_addr);
221 }
222 }
223 }
224
225 contract OwnableData {
226     address public owner;
227     address public pendingOwner;
228 }
229
230 contract Ownable is OwnableData {
231     event OwnershipTransferred(address indexed prev
iousOwner, address indexed newOwner);
232
233     /// @notice `owner` defaults to msg.sender on c
onstruction.
234     constructor() {
235         owner = msg.sender;
236         emit OwnershipTransferred(address(0), msg.s
ender);
237     }
238
239     /// @notice Transfers ownership to `newOwner`.
    Either directly or claimable by the new pending ow
ner.
240     /// Can only be invoked by the current `owner`.
241     /// @param newOwner Address of the new owner.
242     /// @param direct True if `newOwner` should be
set immediately. False if `newOwner` needs to use
`claimOwnership`.
243     /// @param renounce Allows the `newOwner` to be
`address(0)` if `direct` and `renounce` is True. Ha
s no effect otherwise.
244     function transferOwnership(
245         address newOwner,
246         bool direct,
247         bool renounce
248     ) public onlyOwner {
249         if (direct) {
250             // Checks
251             require(newOwner != address(0) || renou
nce, "Ownable: zero address");
252
253             // Effects
254             emit OwnershipTransferred(owner, newOwn
er);
255             owner = newOwner;
256             pendingOwner = address(0);
257         } else {
258             // Effects
259             pendingOwner = newOwner;
260         }
261     }
262
263     /// @notice Needs to be called by `pendingOwner
` to claim ownership.
264     function claimOwnership() public {

```

```

209     bytes memory HEX = "0123456789abcdef";
210     bytes memory _addr = new bytes(42);
211
212     _addr[0] = '0';
213     _addr[1] = 'x';
214
215     for(uint256 i = 0; i < 20; i++) {
216         _addr[2+i*2] = HEX[uint8(_bytes[i + 12]
>> 4)];
217         _addr[3+i*2] = HEX[uint8(_bytes[i + 12]
& 0x0f)];
218     }
219
220     return string(_addr);
221 }
222 }
223 }
224
225 contract OwnableData {
226     address public owner;
227     address public pendingOwner;
228 }
229
230 contract Ownable is OwnableData {
231     event OwnershipTransferred(address indexed prev
iousOwner, address indexed newOwner);
232
233     /// @notice `owner` defaults to msg.sender on c
onstruction.
234     constructor() {
235         owner = msg.sender;
236         emit OwnershipTransferred(address(0), msg.s
ender);
237     }
238
239     /// @notice Transfers ownership to `newOwner`.
    Either directly or claimable by the new pending ow
ner.
240     /// Can only be invoked by the current `owner`.
241     /// @param newOwner Address of the new owner.
242     /// @param direct True if `newOwner` should be
set immediately. False if `newOwner` needs to use
`claimOwnership`.
243     /// @param renounce Allows the `newOwner` to be
`address(0)` if `direct` and `renounce` is True. Ha
s no effect otherwise.
244     function transferOwnership(
245         address newOwner,
246         bool direct,
247         bool renounce
248     ) public onlyOwner {
249         if (direct) {
250             // Checks
251             require(newOwner != address(0) || renou
nce, "Ownable: zero address");
252
253             // Effects
254             emit OwnershipTransferred(owner, newOwn
er);
255             owner = newOwner;
256             pendingOwner = address(0);
257         } else {
258             // Effects
259             pendingOwner = newOwner;
260         }
261     }
262
263     /// @notice Needs to be called by `pendingOwner
` to claim ownership.
264     function claimOwnership() public {

```

```

265     address _pendingOwner = pendingOwner;
266
267     // Checks
268     require(msg.sender == _pendingOwner, "Ownable: caller != pending owner");
269
270     // Effects
271     emit OwnershipTransferred(owner, _pendingOwner);
272
273     owner = _pendingOwner;
274     pendingOwner = address(0);
275 }
276
277 /// @notice Only allows the `owner` to execute the function.
278 modifier onlyOwner() {
279     require(msg.sender == owner, "Ownable: caller is not the owner");
280     _;
281 }
282
283 interface ITreasury {
284     function mintRewards( address _recipient, uint _amount ) external;
285 }
286
287 contract Distributor is Ownable {
288     using LowGasSafeMath for uint;
289     using LowGasSafeMath for uint32;
290
291
292
293     /* ===== VARIABLES ===== */
294
295     IERC20 public immutable TIME;
296     ITreasury public immutable treasury;
297
298     uint32 public immutable epochLength;
299     uint32 public nextEpochTime;
300
301     mapping( uint => Adjust ) public adjustments;
302
303     event LogDistribute(address indexed recipient, uint amount);
304     event LogAdjust(uint initialRate, uint currentRate, uint targetRate);
305     event LogAddRecipient(address indexed recipient, uint rate);
306     event LogRemoveRecipient(address indexed recipient);
307
308     /* ===== STRUCTS ===== */
309
310     struct Info {
311         uint rate; // in ten-thousandths ( 5000 = 0.5% )
312         address recipient;
313     }
314     Info[] public info;
315
316     struct Adjust {
317         bool add;
318         uint rate;
319         uint target;
320     }
321
322
323

```

```

265     address _pendingOwner = pendingOwner;
266
267     // Checks
268     require(msg.sender == _pendingOwner, "Ownable: caller != pending owner");
269
270     // Effects
271     emit OwnershipTransferred(owner, _pendingOwner);
272
273     owner = _pendingOwner;
274     pendingOwner = address(0);
275 }
276
277 /// @notice Only allows the `owner` to execute the function.
278 modifier onlyOwner() {
279     require(msg.sender == owner, "Ownable: caller is not the owner");
280     _;
281 }
282
283 interface ITreasury {
284     function mintRewards( address _recipient, uint _amount ) external;
285 }
286
287 contract Distributor is Ownable {
288     using LowGasSafeMath for uint;
289     using LowGasSafeMath for uint32;
290
291
292
293     /* ===== VARIABLES ===== */
294
295     IERC20 public immutable TIME;
296     ITreasury public immutable treasury;
297
298     uint32 public immutable epochLength;
299     uint32 public nextEpochTime;
300
301     mapping( uint => Adjust ) public adjustments;
302
303     event LogDistribute(address indexed recipient, uint amount);
304     event LogAdjust(uint initialRate, uint currentRate, uint targetRate);
305     event LogAddRecipient(address indexed recipient, uint rate);
306     event LogRemoveRecipient(address indexed recipient);
307
308     /* ===== STRUCTS ===== */
309
310     struct Info {
311         uint rate; // in ten-thousandths ( 5000 = 0.5% )
312         address recipient;
313     }
314     Info[] public info;
315
316     struct Adjust {
317         bool add;
318         uint rate;
319         uint target;
320     }
321
322
323

```

```

324     /* ===== CONSTRUCTOR ===== */
325
326     constructor( address _treasury, address _time,
uint32 _epochLength, uint32 _nextEpochTime ) {
327         require( _treasury != address(0) );
328         treasury = ITreasury(_treasury);
329         require( _time != address(0) );
330         TIME = IERC20(_time);
331         epochLength = _epochLength;
332         nextEpochTime = _nextEpochTime;
333     }
334
335
336
337     /* ===== PUBLIC FUNCTIONS ===== */
338
339     /**
340     @notice send epoch reward to staking contra
ct
341     */
342     function distribute() external returns ( bool )
{
343         if ( nextEpochTime <= uint32(block.timestam
p) ) {
344             nextEpochTime = nextEpochTime.add32( ep
ochLength ); // set next epoch time
345
346             // distribute rewards to each recipient
347             for ( uint i = 0; i < info.length; i++
) {
348                 if ( info[ i ].rate > 0 ) {
349                     treasury.mintRewards( // mint a
nd send from treasury
350                         info[ i ].recipient,
351                         nextRewardAt( info[ i ].rat
e )
352                     );
353                     adjust( i ); // check for adjus
tment
354                 }
355                 emit LogDistribute(info[ i ].recipi
ent, nextRewardAt( info[ i ].rate ));
356             }
357             return true;
358         } else {
359             return false;
360         }
361     }
362
363
364
365     /* ===== INTERNAL FUNCTIONS ===== */
366
367     /**
368     @notice increment reward rate for collector
369     */
370     function adjust( uint _index ) internal {
371         Adjust memory adjustment = adjustments[ _in
dex ];
372         if ( adjustment.rate != 0 ) {
373             uint initial = info[ _index ].rate;
374             uint rate = initial;
375             if ( adjustment.add ) { // if rate shou
ld increase
376                 rate = rate.add( adjustment.rate );
// raise rate
377                 if ( rate >= adjustment.target ) {
// if target met

```

```

324     /* ===== CONSTRUCTOR ===== */
325
326     constructor( address _treasury, address _time,
uint32 _epochLength, uint32 _nextEpochTime ) {
327         require( _treasury != address(0) );
328         treasury = ITreasury(_treasury);
329         require( _time != address(0) );
330         TIME = IERC20(_time);
331         epochLength = _epochLength;
332         nextEpochTime = _nextEpochTime;
333     }
334
335
336
337     /* ===== PUBLIC FUNCTIONS ===== */
338
339     /**
340     @notice send epoch reward to staking contra
ct
341     */
342     function distribute() external returns ( bool )
{
343         if ( nextEpochTime <= uint32(block.timestam
p) ) {
344             nextEpochTime = nextEpochTime.add32( ep
ochLength ); // set next epoch time
345
346             // distribute rewards to each recipient
347             for ( uint i = 0; i < info.length; i++
) {
348                 if ( info[ i ].rate > 0 ) {
349                     treasury.mintRewards( // mint a
nd send from treasury
350                         info[ i ].recipient,
351                         nextRewardAt( info[ i ].rat
e )
352                     );
353                     adjust( i ); // check for adjus
tment
354                 }
355                 emit LogDistribute(info[ i ].recipi
ent, nextRewardAt( info[ i ].rate ));
356             }
357             return true;
358         } else {
359             return false;
360         }
361     }
362
363
364
365     /* ===== INTERNAL FUNCTIONS ===== */
366
367     /**
368     @notice increment reward rate for collector
369     */
370     function adjust( uint _index ) internal {
371         Adjust memory adjustment = adjustments[ _in
dex ];
372         if ( adjustment.rate != 0 ) {
373             uint initial = info[ _index ].rate;
374             uint rate = initial;
375             if ( adjustment.add ) { // if rate shou
ld increase
376                 rate = rate.add( adjustment.rate );
// raise rate
377                 if ( rate >= adjustment.target ) {
// if target met

```

```

378         rate = adjustment.target;
379         delete adjustments[ _index ];
380     }
381     } else { // if rate should decrease
382         rate = rate.sub( adjustment.rate );
    // lower rate
383     if ( rate <= adjustment.target ) {
    // if target met
384         rate = adjustment.target;
385         delete adjustments[ _index ];
386     }
387     }
388     info[ _index ].rate = rate;
389     emit LogAdjust(initial, rate, adjustment.target);
390 }
391 }
392
393
394
395 /* ===== VIEW FUNCTIONS ===== */
396 /**
397  *
398  * @notice view function for next reward at given rate
399  * @param _rate uint
400  * @return uint
401  */
402 function nextRewardAt( uint _rate ) public view returns ( uint ) {
403     return TIME.totalSupply().mul( _rate ).div( 1000000 );
404 }
405
406 /**
407  * @notice view function for next reward for specified address
408  * @param _recipient address
409  * @return uint
410  */
411 function nextRewardFor( address _recipient ) external view returns ( uint ) {
412     uint reward;
413     for ( uint i = 0; i < info.length; i++ ) {
414         if ( info[ i ].recipient == _recipient ) {
415             reward = nextRewardAt( info[ i ].rate );
416         }
417     }
418     return reward;
419 }
420
421
422
423 /* ===== POLICY FUNCTIONS ===== */
424 /**
425  *
426  * @notice adds recipient for distributions
427  * @param _recipient address
428  * @param _rewardRate uint
429  */
430 function addRecipient( address _recipient, uint _rewardRate ) external onlyOwner {
431     require( _recipient != address(0), "IA" );
432     require( _rewardRate <= 5000, "Too high reward rate");

```

```

378         rate = adjustment.target;
379         delete adjustments[ _index ];
380     }
381     } else { // if rate should decrease
382         rate = rate.sub( adjustment.rate );
    // lower rate
383     if ( rate <= adjustment.target ) {
    // if target met
384         rate = adjustment.target;
385         delete adjustments[ _index ];
386     }
387     }
388     info[ _index ].rate = rate;
389     emit LogAdjust(initial, rate, adjustment.target);
390 }
391 }
392
393
394
395 /* ===== VIEW FUNCTIONS ===== */
396 /**
397  *
398  * @notice view function for next reward at given rate
399  * @param _rate uint
400  * @return uint
401  */
402 function nextRewardAt( uint _rate ) public view returns ( uint ) {
403     return TIME.totalSupply().mul( _rate ).div( 1000000 );
404 }
405
406 /**
407  * @notice view function for next reward for specified address
408  * @param _recipient address
409  * @return uint
410  */
411 function nextRewardFor( address _recipient ) external view returns ( uint ) {
412     uint reward;
413     for ( uint i = 0; i < info.length; i++ ) {
414         if ( info[ i ].recipient == _recipient ) {
415             reward = nextRewardAt( info[ i ].rate );
416         }
417     }
418     return reward;
419 }
420
421
422
423 /* ===== POLICY FUNCTIONS ===== */
424 /**
425  *
426  * @notice adds recipient for distributions
427  * @param _recipient address
428  * @param _rewardRate uint
429  */
430 function addRecipient( address _recipient, uint _rewardRate ) external onlyOwner {
431     require( _recipient != address(0), "IA" );
432     require( _rewardRate <= 50000, "Too high reward rate");

```



```

433         require(info.length <= 4, "limit recipients
max to 5");
434         info.push( Info({
435             recipient: _recipient,
436             rate: _rewardRate
437         }));
438         emit LogAddRecipient(_recipient, _rewardRate);
439     }
440
441     /**
442      * @notice removes recipient for distributions
443      * @param _index uint
444      * @param _recipient address
445      */
446     function removeRecipient( uint _index, address
_recipient ) external onlyOwner {
447         require( _recipient == info[ _index ].recipient, "NA" );
448         info[_index] = info[info.length-1];
449         adjustments[_index] = adjustments[ info.length-1 ];
450         info.pop();
451         delete adjustments[ info.length-1 ];
452         emit LogRemoveRecipient(_recipient);
453     }
454
455     /**
456      * @notice set adjustment info for a collector's reward rate
457      * @param _index uint
458      * @param _add bool
459      * @param _rate uint
460      * @param _target uint
461      */
462     function setAdjustment( uint _index, bool _add,
uint _rate, uint _target ) external onlyOwner {
463         require(_target <= 5000, "Too high reward rate");
464         adjustments[ _index ] = Adjust({
465             add: _add,
466             rate: _rate,
467             target: _target
468         });
469     }
470 }

```

```

433         require(info.length <= 4, "limit recipients
max to 5");
434         info.push( Info({
435             recipient: _recipient,
436             rate: _rewardRate
437         }));
438         emit LogAddRecipient(_recipient, _rewardRate);
439     }
440
441     /**
442      * @notice removes recipient for distributions
443      * @param _index uint
444      * @param _recipient address
445      */
446     function removeRecipient( uint _index, address
_recipient ) external onlyOwner {
447         require( _recipient == info[ _index ].recipient, "NA" );
448         info[_index] = info[info.length-1];
449         adjustments[_index] = adjustments[ info.length-1 ];
450         info.pop();
451         delete adjustments[ info.length-1 ];
452         emit LogRemoveRecipient(_recipient);
453     }
454
455     /**
456      * @notice set adjustment info for a collector's reward rate
457      * @param _index uint
458      * @param _add bool
459      * @param _rate uint
460      * @param _target uint
461      */
462     function setAdjustment( uint _index, bool _add,
uint _rate, uint _target ) external onlyOwner {
463         require(_target <= 50000, "Too high reward rate");
464         adjustments[ _index ] = Adjust({
465             add: _add,
466             rate: _rate,
467             target: _target
468         });
469     }
470 }

```