

<libs>
SafeMath
#mul(a:uint,b:uint): uint const
#div(a:uint,b:uint): uint const
#sub(a:uint,b:uint): uint const
#add(a:uint,b:uint): uint const

Crowdsale
+token: ERC20
+rate: uint
+rateRaised: uint
-<event>> TokenPurchase(purchase:address, beneficiary:address, value:uint,amount:uint)
+Crowdsale(_rate:uint,_wallet:address,_token:ERC20) payable
+buyTokens(_beneficiary:address) payable
#_preValidatePurchase(_beneficiary:address,_weiAmount:uint)
#_postValidatePurchase(_beneficiary:address,_weiAmount:uint)
#_deliverTokens(_beneficiary:address,_tokenAmount:uint)
#_processPurchase(_beneficiary:address,_tokenAmount:uint)
#_updatePurchasingState(_beneficiary:address,_weiAmount:uint)
#_getTokenAmount(_weiAmount): uint const
#_forwardFunds()

CappedCrowdsale
+cap: uint
+CappedCrowdsale(_cap:uint)
+capReached(): bool const
#_preValidatePurchase(_beneficiary:address,_weiAmount:uint)

TimedCrowdsale
+openingTime: uint
+closingTime: uint
-<modifier>> onWhirlwindOpen()
+TimedCrowdsale(_openingTime:uint,_closingTime:uint)
#_preValidatePurchase(_beneficiary:address,_weiAmount:uint) onlyWhirlwindOpen

FinalizableCrowdsale
+isFinalized: bool = false
-<event>> Finalized()
+finalize() onlyOwner
#finalize()

RefundableCrowdsale
+goal: uint
-escrow: RefundEscrow
+RefundableCrowdsale(_goal:uint)
+claimRefund()
+goalReached(): bool const
#_forwardFunds()

ConditionalEscrow
+withdrawAllowed(_payee:address): bool const
+withdraw(_payee:address)

RefundEscrow
+state: State
+beneficiary: address
-<event>> Closed()
+constructor(_beneficiary:address)
+deposit(_refund:address)
+lose() onlyOwner
+enableRefunds() onlyOwner
+beneficiaryWithdrawal(investor:address)
+withdrawAllowed(_payee:address): bool const

<<enum>>
State
Active
Refunding
Closed

ERC20
-<event>> Approval(owner:address,spender:address,value:uint)
+allowance(owner:address,spender:address): uint const
+transferFrom(address,spender,value:uint): bool
+approve(spender:address,value:uint): bool

ERC20Basic
-<event>> Transfer(from:address,to:address,value:uint)
+totalSupply(): uint const
+balanceOf(address): uint const
+transfer(to:address,value:uint): bool

Ownable
+owner: address
-<event>> OwnershipTransferred(owner:address,newOwner:address)
+modifier>> onlyOwner()
+Ownable()
+transferOwnership(newOwner:address) onlyOwner

ProfitSharing
+accounts: address[] InvestorAccount
+profitDepositor: address
+totalProfits: uint
+totalSupplyFixed: bool
#totalSupply: uint
-<event>> ProfitDepositorChanged(newProfitDepositor:address)
-<event>> ProfitDeposited(deposit:address, amount:uint)
-<event>> ProfitShareUpdated(investor:address, amount:uint)
-<event>> ProfitWithdrawal(investor:address, amount:uint)
-<modifier>> onlyProfitDepositor()
+constructor(_profitDepositor:address)
+setProfitDepositor(_newProfitDepositor:address) onlyOwner
+depositProfit() payable onlyProfitDepositor
+profitSharing(_investor:address): uint const
+updateProfitShare(_investor:address)
+withdrawProfitShare()

<<struct>>
InvestorAccount
+balance: uint
+lastTotalProfits: uint
+profitShare: uint

Whitelisted
+whitelist: Whitelist
-<event>> WhitelistChanged(newWhitelist:address)
-<modifier>> onlyWhitelisted(_address:address)
+constructor(_whitelist:Whitelist)
+setWhitelist(_newWhitelist:Whitelist) onlyOwner

Whitelist
+admins: (address,bool)
+isWhitelisted: (address,bool)
-<event>> AdminAdded(admin:address)
-<event>> AdminRemoved(admin:address)
-<event>> InvestorAdded(admin:address, investor:address)
-<event>> InvestorRemoved(admin:address, investor:address)
-<modifier>> onlyAdmin()
+addAdmin(_admin:address) onlyOwner
+removeAdmin(_admin:address) onlyOwner
+addWhitelisted(_investors:address[*]) onlyAdmin
+removeFromWhitelist(_investors:address[*]) onlyAdmin

