

<<library>>

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

Ownable

+owner: address

<<event>> OwnershipTransferred(owner:address,

newOwner:address)

<<modifier>> onlyOwner()

+disable()

+transferOwnership(newOwner:address)

onlyOwner

ERC20Basic

<<event>> Transfer(from:address,to:address,

value:uint)

+totalSupply(): uint const

+balanceOf(who:address): uint const

+transfer(to:address,value:uint): bool

ERC20

<<event>> Approval(owner:address,spender:address,

value:uint)

+allowance(owner:address,spender:address): uint const

+approve(spender:address,value:uint): bool

+approveFrom(msg.sender,value:uint): bool

Escrow

-deposits: {address:uint}

<<event>> Deposited(payee:address,weiAmount:uint)

<<event>> Withdrawn(payee:address,weiAmount:uint)

+depositOf(_payee:address): uint const

+deposit(_payee:address)

payable

onlyOwner

+withdraw(_payee:address)

onlyOwner

ConditionalEscrow

+withdrawalAllowed(_payee:address): bool const

+withdraw(_payee:address)

RefundEscrow

+state: State

+beneficiary: address

<<event>> Closed()

<<event>> RefundEnabled()

+constructor(_beneficiary:address)

+deposit(_refundee:address)

payable

+close()

onlyOwner

+enableRefunds()

onlyOwner

+beneficiaryWithdraw()

+withdrawalAllowed(_payee:address): bool const

<<enum>>

State

Active

Refunding

Closed

Crowdsale

+token: ERC20

+wallet: address

+rate: uint

+weiRaised: uint

<<event>> TokenPurchase(purchaser:address,

beneficiary:address,

value:uint,amount:uint)

+Crowdsale(_rate:uint,_wallet:address,_oken:ERC20)

payable

+()

+buyTokens(_beneficiary:address)

payable

#_preValidatePurchase(_beneficiary:address,

_weiAmount:uint)

#_postValidatePurchase(_beneficiary:address,

_weiAmount:uint)

#_deliverTokens(_beneficiary:address,_tokenAmount:uint)

#_updatePurchaseState(_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>> onlyWhileOpen()

+TimedCrowdsale(openingTime:uint,_closingTime:uint)

+hasClosed(): bool const

*_preValidatePurchase(_beneficiary:address,

_weiAmount:uint)

onlyWhileOpen

FinalizableCrowdsale

+isFinalized: bool = false

<<event>> Finalized()

+finalize()

onlyOwner

#finalization()

RefundableCrowdsale

+goal: uint

-escrow: RefundEscrow

+RefundableCrowdsale(goal:uint)

+claimRefund()

+goalReached(): bool const

#finalization()

#_forwardFunds()

ProfitSharing

+accounts: {address-InvestorAccount}

+profitDepositor: address

+totalProfits: uint

+totalSupplyFixed: bool

#totalSupply: uint

<<event>> ProfitDepositorChanged(newProfitDepositor:address)

<<event>> ProfitShareUpdated(investor:address,

amount:uint)

<<event>> ProfitWithdrawal(investor:address,

amount:uint)

<<modifier>> onlyProfitDepositor()

+constructor(_profitDepositor:address)

payable

+setProfitDepositor(_newProfitDepositor:address)

onlyOwner

+depositProfit()

payable

onlyProfitDepositor

+profitShareOfing(_investor:address): uint const

+updateProfitShare(_investor:address)

withdrawProfitShare()

<<struct>>

InvestorAccount

balances: uint

lastTotalProfits: uint

+profitShare: uint

Whitelist

+admins: {address-bool}

+isWhitelisted: {address-bool}

<<event>> AdminAdded(admin:address)

<<modifier>> onlyWhitelisted(address:address)

+constructor(_whitelist:Whitelist)

+setWhitelist(_newWhitelist:Whitelist)

onlyOwner

<<modifier>> onlyAdmin()

+addAdmin(_admin:address)

onlyOwner

+removeAdmin(_admin:address)

onlyOwner

+addToWhitelist(_investors:address(*))

onlyAdmin

+removeFromWhitelist(_investors:address(*))

onlyAdmin

Whitelisted

+whitelist: Whitelist

<<event>> WhitelistChanged(newWhitelist:address)

<<modifier>> onlyWhitelisted(address:address)

+constructor(_whitelist:Whitelist)

+setWhitelist(_newWhitelist:Whitelist)

onlyOwner

