

<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, allowance:address, spender:address): uint const	
+allowance(owner:address, spender:address): uint const	
+transferFrom(from:address, to:address, value:uint): bool	
+approve(spender:address,value:uint): bool	

Escrow	
-deposits: {address->uint}	
<<event>> Deposited(payee:address,weiAmount:uint)	
<<event>> Withdrawn(payee:address,weiAmount:uint)	
-depositsOf(_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>> RefundsEnabled()	
<<event>> RefundsDisabled()	
+close() payable	
+deposit(_refundee:address) payable onlyOwner	
+enableRefunds() onlyOwner	
+beneficiaryWithdraw(investor:address) payable	
+withdrawalAllowed(_payee:address): bool const	

State	
<<enum>>	
Active	
Refunding	
Closed	

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()	

Crowdsale	
+token: ERC20	
+wallet: address	
+rate: uint	
+weiRaised: uint	
<<event>> TokenPurchase(purchaser:address, beneficiary:address, value:uint,amount:uint)	
+Crowdsale(_rate:uint,_wallet:address,_token:ERC20) payable	
+() 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)	

ProfitSharing	
-accounts: {address->InvestorAccount}	
+profitShareTo: address	
+totalProfitShare: uint	
+totalSupplyFixed: bool	
#totalSupply: uint	
<<event>> ProfitDeposited(depositor:address, amount:uint)	
<<event>> ProfitShareUpdated(investor:address, amount:uint)	
<<event>> ProfitWithdrawal(investor:address, amount:uint)	
<<modifier>> onlyProfitDepositor()	
+constructor(_profitDepositor:address) onlyOwner	
+setProfitDepositor(_newProfitDepositor:address) payable	
+depositProfit() payable onlyProfitDepositor	
+profitSharing(_investor:address): uint const	
+profitShareTo: address	
+withdrawProfitShare()	

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

Whitelisted	
-whitelist: Whitelist	
<<event>> WhitelistChanged(newWhitelist:address)	
<<modifier>> onlyWhitelisted(_address:address)	
+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	
+addToWhitelist(_investors:address*) onlyAdmin	
+removeFromWhitelist(_investors:address*) onlyAdmin	

MintableToken	
+minter: address	

StorxCrowdsale	
+tokenCap: uint	

KeyRecoverable	
+keyRecoverer: KeyRecoverer	

KeyRecoverer	
+indices: {address->uint}	

