SolanaWallet

CODERS: Riccardo Torrisi, Federico Arona, Daniele Calanna

USolanaWallet Class Reference

`#include < SolanaWallet.h >`

Public Member Functions

void	SetSaveSlotName (FString NewSaveSlotName)
const FString &	**GetSaveSlotName** () const
bool	**DoesWalletExist** () const
bool	**GenerateMnemonic** (FString &MnemonicString)
bool	**RestoreMnemonic** (FString InMnemonic)
FString	**GetMnemonicString** () const
	DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam (FOnMnemonicUpdated, FString, Mnemonic)
bool	**SetPassword** (FString NewPassword)
bool	**SaveWallet** ()
bool	**UnlockWallet** (FString Password)
void	**LockWallet** (bool bSaveWallet)
void	**WipeWallet** ()
	DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam (FOnWalletWiped, **USolanaWallet** *, Wallet)
bool	**IsWalletLocked** () const
bool	**SetDerivationPath** (const **FDerivationPath** &DerivationPath)
bool	**GetAccountsFromPath** (**FDerivationPath** Path, int32 NumAccounts, TArray< FAccount > &OutAccounts) const
UWalletAccount *	**GetAccountFromGenIndex** (int32 GenIndex) const
UWalletAccount *	**GenerateAccountFromGenIndex** (int32 GenIndex)
uint32	**GetNextAccountIndexToGenerate** () const
UWalletAccount *	**GenerateNewAccount** ()
UWalletAccount *	**ImportAccountFromPrivateKey** (FString PrivateKey)
UWalletAccount *	**ImportAccountFromPublicKey** (FString PublicKey)
void	**RemoveAccount** (**UWalletAccount** *Account)
TArray< **UWalletAccount** * >	**GetAccounts** () const

Static Public Member Functions

static bool	IsMnemonicValid (FString Mnemonic)
static TArray< **FDerivationPath** >	**GetDerivationPaths** ()
static void	**ClipboardCopy** (FString String)

Public Attributes

FOnMnemonicUpdated	OnMnemonicUpdated
FOnWalletWiped	**OnWalletWiped**

Detailed Description

USolanaWallet

This class abstract a wallet for the solana network and it is made up of:

- a mnemonic phrase to generate new accounts;
- a derivation path to generate new accounts;
- a save slot name to save the wallet on disk;
- a password to encrypt the wallet on disk;
- a list of accounts either generated from the mnemonic phrase or imported from a public or private key;

Member Function Documentation

◆ ClipboardCopy()

static void USolanaWallet::ClipboardCopy(FString *String*)static

Copy the string parameter to the system clipboard.

ParametersStringThe string to copy.

◆ DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam() [1/2]

USolanaWallet::DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnMnemonicUpdated ,FString ,Mnemonic)

Called when mnemonic is set, loaded or erased.

ParametersMnemonicThe Updated Mnemonic

◆ DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam() [2/2]

USolanaWallet::DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnWalletWiped ,USolanaWallet * ,Wallet)

Called when the wallet get wiped;

ParametersWalletThe wiped wallet

◆ DoesWalletExist()

bool USolanaWallet::DoesWalletExist()const

Check if there is an existing file for this wallet.

ReturnsWhether the wallet file already exists or not.

◆ GenerateAccountFromGenIndex()

UWalletAccount * USolanaWallet::GenerateAccountFromGenIndex(int32 *GenIndex*)

Generate an account with the given generation index.

ParametersGenIndexThe generation index.

ReturnsThe generated **UWalletAccount**.

◆ GenerateMnemonic()

bool USolanaWallet::GenerateMnemonic(FString & *MnemonicString*)

Generate a mnemonic if no mnemonic exists in this wallet.

ParametersMnemonicStringReturn the mnemonic string currently in use.

ReturnsWhether the mnemonic has been generated or not.

◆ GenerateNewAccount()

UWalletAccount * USolanaWallet::GenerateNewAccount()

Generate a new account increasing the generation index.

ReturnsThe generated account.

◆ GetAccountFromGenIndex()

UWalletAccount * USolanaWallet::GetAccountFromGenIndex(int32 *GenIndex*)const

Get the account corresponding to the given generation index if it has been already generated.

ParametersGenIndexThe generation index.

ReturnsThe corresponding **UWalletAccount**.

◆ GetAccounts()

TArray< UWalletAccount * > USolanaWallet::GetAccounts()const

Get all accounts in this wallet.

ReturnsThe list of account for this wallet.

◆ GetAccountsFromPath()

bool USolanaWallet::GetAccountsFromPath(FDerivationPath *Path*,int32 *NumAccounts*,TArray< FAccount > & *OutAccounts*)const

Get accounts for a specific derivation path.

ParametersPathThe DerivationPath.**NumAccounts**The number of accounts to retrieve.**OutAccounts**The list of accounts

ReturnsWhether the accounts were found for the given derivation path.

◆ GetDerivationPaths()

static TArray< FDerivationPath > USolanaWallet::GetDerivationPaths()static

Get all available derivation paths.

ReturnsThe list of available derivation paths.

◆ GetMnemonicString()

FString USolanaWallet::GetMnemonicString()const

Get the Mnemonic string of this wallet.

ReturnsThe mnemonic of this wallet.

◆ GetNextAccountIndexToGenerate()

uint32 USolanaWallet::GetNextAccountIndexToGenerate()const

Get the index of the next account to generate.

ReturnsThe index of the next account to generate.

◆ GetSaveSlotName()

const FString & USolanaWallet::GetSaveSlotName()constinline

Get the name of the file used to load or save this wallet.

ReturnsName of the slot name file currently in use.

◆ ImportAccountFromPrivateKey()

UWalletAccount * USolanaWallet::ImportAccountFromPrivateKey(FString *PrivateKey*)

Create an account from a private key.

ParametersPrivateKeyThe private key.

ReturnsThe created account.

◆ ImportAccountFromPublicKey()

UWalletAccount * USolanaWallet::ImportAccountFromPublicKey(FString *PublicKey*)

Create an account from a public key.

ParametersPublicKeyThe public key.

ReturnsThe created account.

◆ IsMnemonicValid()

static bool USolanaWallet::IsMnemonicValid(FString *Mnemonic*)static

Check if a Mnemonic string is valid.

ParametersMnemonicThe Mnemonic to check

ReturnsWhether the mnemonic is valid or not.

◆ IsWalletLocked()

bool USolanaWallet::IsWalletLocked()const

Whether the wallet is locked or not.

ReturnsWhether the wallet is locked or not.

◆ LockWallet()

void USolanaWallet::LockWallet(bool *bSaveWallet*)

Lock the wallet, deleting mnemonic and private keys from memory.

ReturnsWhether the lock was successful or not.

◆ RemoveAccount()

void USolanaWallet::RemoveAccount(UWalletAccount * *Account*)

Remove an account from this wallet.

ParametersAccountThe account to remove.

◆ RestoreMnemonic()

bool USolanaWallet::RestoreMnemonic(FString *InMnemonic*)

Restore a mnemonic if no mnemonic exists in this wallet.

ParametersInMnemonicThe new Mnemonic

ReturnsWhether the mnemonic has been restored or not.

◆ SaveWallet()

bool USolanaWallet::SaveWallet()

Save this wallet to disk to reload it later.

ReturnsWhether the save was successful or not.

◆ SetDerivationPath()

bool USolanaWallet::SetDerivationPath(const FDerivationPath & *DerivationPath*)

Set the derivation path for this wallet to derive new wallet address.

ParametersDerivationPathThe new DerivationPath.

ReturnsWhether the new DerivationPath has been set or not.

◆ SetPassword()

bool USolanaWallet::SetPassword(FString *NewPassword*)

Set or change the password.

ParametersNewPasswordThe new password.

ReturnsWhether the new password has been set or not.

◆ SetSaveSlotName()

void USolanaWallet::SetSaveSlotName(FString *NewSaveSlotName*)

Set the name of the file used to load or save this wallet.

ParametersNewSaveSlotNameName of the slot name file to use.

◆ UnlockWallet()

bool USolanaWallet::UnlockWallet(FString *Password*)

Load and unlock this wallet from disk if password is correct.

ReturnsWhether the unlock was successful or not.

◆ WipeWallet()

void USolanaWallet::WipeWallet()

Wipe the wallet from both memory and disk.

ReturnsWhether the wipe was successful or not.

SolanaWalletManager

USolanaWalletManager Class Reference

Public Member Functions

virtual void	Initialize (FSubsystemCollectionBase &Collection) override
TArray< FString >	**GetSaveSlotList** () const
USolanaWallet *	**CreateNewWallet** ()
USolanaWallet *	**GetOrCreateWallet** (const FString &SlotName)
void	**RegisterWallet** (**USolanaWallet** *Wallet)

Member Function Documentation

◆ CreateNewWallet()

USolanaWallet * USolanaWalletManager::CreateNewWallet()

Create a new wallet.

ReturnsThe created **USolanaWallet**.

◆ GetOrCreateWallet()

USolanaWallet * USolanaWalletManager::GetOrCreateWallet(const FString & *SlotName*)

Get a wallet from a slot name, create a new one if not exists.

ParametersSlotNameThe slot name.

ReturnsThe created or retrieved account.

◆ GetSaveSlotList()

TArray< FString > USolanaWalletManager::GetSaveSlotList()const

Get the list of available save slots.

ReturnsThe array of available slots.

◆ RegisterWallet()

void USolanaWalletManager::RegisterWallet(USolanaWallet * *Wallet*)

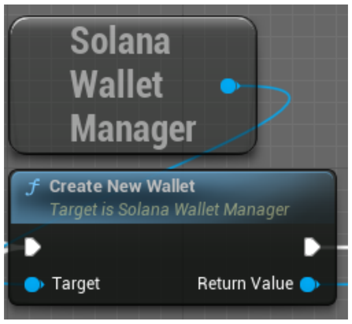
Register a newly created wallet into the list of wallets.

ParametersWalletThe private key.

How To - A practical approach

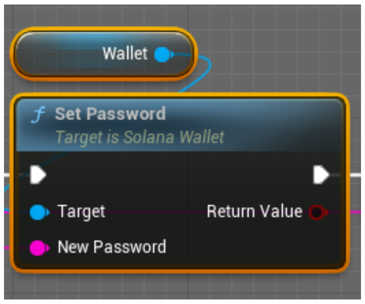
Create Wallet flow

The flow to create a new wallet is preatty straight forward.

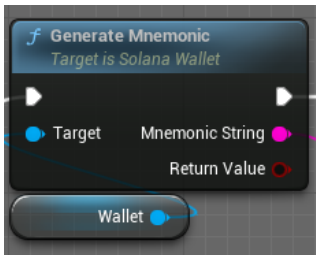


It is enough to call the "*CreateNewWallet*" function from the SolanaWalletManager.

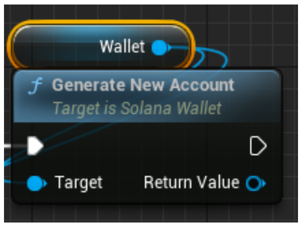
From the returned Wallet object it is mandatory to set a new password: this can be done with the "*SetPassword*" function.



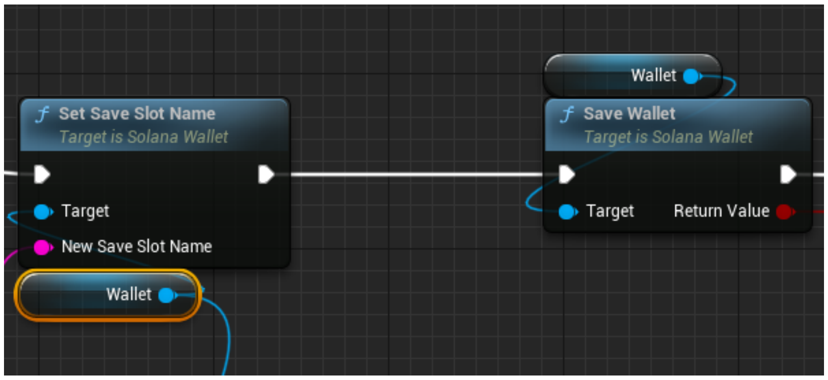
Then it is mandatory to generate a Mnemonic for the given Wallet. This can be done with the "*GenerateMnemonic*" function. The Sting returned is the "SeedPhrase" that can be used to restore the wallet.



The new Wallet needs at least an account. An account can be generated with the "*GenerateNewAccount*" function.



Finally, the created wallet can be saved to the local system. This is done with the "*SetSaveSlotName*" and "*SaveWallet*" functions. The SaveSlotName parameter is suggested to be a combination of the wallet name and the public key of the wallet.



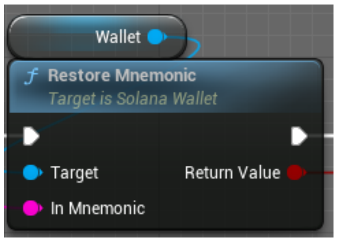
Recover Wallet flow

A wallet can be recovered both with “PrivateKey” or “SeedPhrase”.

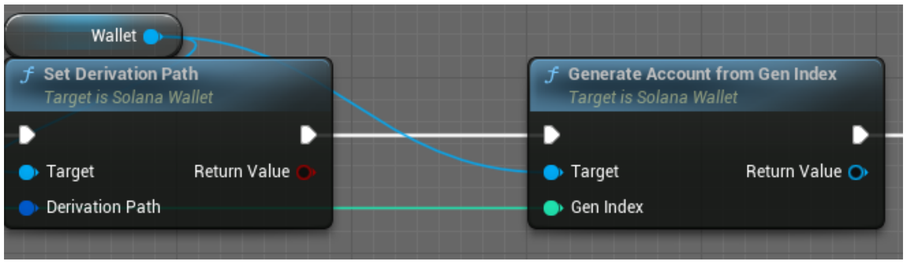
In order to restore a wallet you need to create a new Wallet object as shown in the first step of the “Create Wallet Flow”.

Restore from seed phrase

From the created Wallet object, you need to restore the Mnemonic



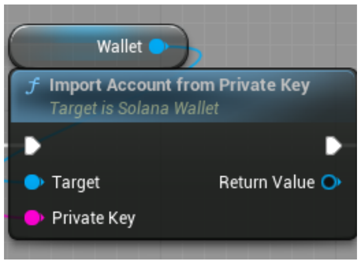
Then, in order to retrieve the accounts, a derivation path must be selected



Then it is needed to set a new password for the wallet and save it as seen in the “Create Wallet” flow.

Restore from private key

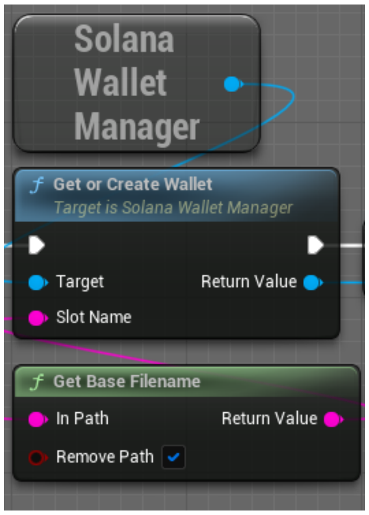
From the created wallet object, it is enough to call the “*ImportAccountFromPrivateKey*”

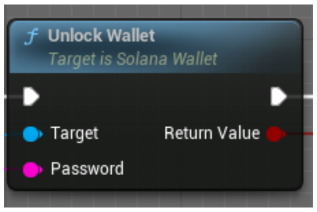


Then it is needed to set a new password for the wallet and save it as seen in the “Create Wallet” flow.

Unlock an existing wallet

In order to login to an existing wallet it is enought to retrieve the existing wallet from a “SaveSlotName” using the “*GetOrCreateWallet*” and then calll the “*UnlockWallet*” function providing the correct password.


Solana Wallet Manager
Get or Create Wallet
Target is Solana Wallet Manager
Target
Return Value
Slot Name
Get Base Filename
In Path
Return Value
Remove Path


Unlock Wallet
Target is Solana Wallet
Target
Return Value
Password