Commandes de l'explorateur Bitcoin (bx)

2022-03-17

Bitcoin Explorer (bx) est un outil en ligne de commande qui offre une variété de commandes pour la gestion des clés et la construction de transactions. Il fait partie de la bibliothèque libbitcoin bitcoin.

Utilisation : bx COMMANDE [--help]  
  
Info : Les commandes bx sont :  
  
adresse-décoder  
adresse intégrée  
adresse-encoder  
adresse-valider  
décodage en base16  
encoder en base16  
décodage base58  
encoder en base58  
base58check-décoder  
base58check-encoder  
décodage base64  
encoder en base64  
Bitcoin160  
Bitcoin256  
btc-à-satoshi  
ec-add  
ec-add-secrets  
ec-multiplier  
ec-multiplier-secrets  
ec-nouveau  
ec-to-address  
ec-to-public  
ec vers wif  
récupérer le solde  
récupérer l&#39;en-tête  
hauteur de récupération  
récupérer l&#39;historique  
aller chercher furtivement  
récupérer-tx  
récupérer-tx-index  
hd-nouveau  
hd-privé  
hd-public  
hd-to-address  
hd-vers-ec  
hd-to-public  
hd-vers-wif  
aider  
jeu d&#39;entrées  
signe d&#39;entrée  
entrée-valider  
message-sign  
message-valider  
décodage mnémonique  
mnemonic-encode  
mûrmd160  
satoshi-to-btc  
décodage de script  
encodage de script  
script vers adresse  
planter  
envoyer-tx  
envoyer-tx-nœud  
envoyer-tx-p2p  
réglages  
sha160  
sha256  
sha512  
décodage furtif  
encodage furtif  
furtif-public  
secret-furtif  
furtivement partagé  
décodage tx  
tx-encoder  
décodage uri  
uri-encoder  
valider-tx  
adresse de la montre  
wif-à-ec  
wif-au-public  
décodage enveloppant  
envelopper-encoder

Pour plus d'informations, consultez la [Bitcoin Explorer homepage](https://github.com/libbitcoin/libbitcoin-explorer) et [Bitcoin Explorer user documentation](https://github.com/libbitcoin/libbitcoin-explorer/wiki).

# Exemples d'utilisation de la commande bx

Regardons quelques exemples d'utilisation des commandes Bitcoin Explorer pour expérimenter avec des clés et des adresses.

Générez une valeur "seed" aléatoire à l'aide de la commande seed, qui utilise le générateur de nombres aléatoires du système d'exploitation. Transmettez la graine à la commande ec-new pour générer une nouvelle clé privée. Nous sauvegardons la sortie standard dans le fichier *private\_key* :

$ bx semence | bx ec-new &gt; clé\_privée  
$ chat clé\_privée  
73096ed11ab9f1db6135857958ece7d73ea7c30862145bcc4bbc7649075de474

Maintenant, générez la clé publique à partir de cette clé privée à l'aide de la commande ec-to-public. Nous passons le fichier *private\_key* dans l'entrée standard et enregistrons la sortie standard de la commande dans un nouveau fichier *public\_key* :

$ bx ec-to-public &lt; clé\_privée &gt; clé\_publique  
$ cat public\_key  
02fca46a6006a62dfdd2dbb2149359d0d97a04f430f12a7626dd409256c12be500

Nous pouvons reformater la public\_key en tant qu'adresse à l'aide de la commande ec-to-address. Nous passons la *clé\_publique* dans l'entrée standard :

$ bx ec-to-address &lt; public\_key  
17re1S4Q8ZHyCP8Kw7xQad1Lr6XUzWUnkG

Les clés générées de cette manière produisent un portefeuille non déterministe de type 0. Cela signifie que chaque clé est générée à partir d'une graine indépendante. Les commandes de Bitcoin Explorer peuvent également générer des clés de manière déterministe, conformément au BIP-32. Dans ce cas, une clé "maître" est créée à partir d'une graine, puis étendue de manière déterministe pour produire un arbre de sous-clés, résultant en un portefeuille déterministe de type 2.

Tout d'abord, nous utilisons les commandes seed et hd-new pour générer une clé principale qui servira de base pour dériver une hiérarchie de clés :

$ bx graine &gt; graine  
$ graines de chat  
eb68ee9f3df6bd4441a9feadec179ff1  
  
$ bx hd-new &lt; graine &gt; maître  
$ maître de chat  
xprv9s21ZrQH143K2BEhMYpNQouvAgiEjArAVaZaCTgsaGe6LsAnwubeiTcDzd23mAoyizm9cApe51gNfLMkBqkYoWWMCRwzfuJk8RwF1SVEpAQ

Nous utilisons maintenant la commande hd-private pour générer une clé de "compte" renforcée et une séquence de deux clés privées au sein du compte :

$ bx hd-private --hard &lt; maître &gt; compte  
compte chat en $  
xprv9vkDLt81dTKjwHB8fsVB5QK8cGnzveChzSrtCfvu3aMWvQaThp59ueufuyQ8Qi3qpjk4aKsbmbfxwcgS8PYbgoR2NWHeLyvg4DhoEE68A1n  
  
$ bx hd-privé --index 0 &lt; compte  
xprv9xHfb6w1vX9xgZyPNXVgAhPxSsEkeRcPHEUV5iJcVEsuUEACvR3NRY3fpGhcnBiDbvG4LgndirDsia1e9F3DWPkX7Tp1V1u97HKG1FJwUpU  
  
$ bx hd-privé --index 1 &lt; compte  
xprv9xHfb6w1vX9xjc8XbN4GN86jzNAZ6xHEqYxzbLB4fzHFd6VqCLPGRZFsdjsuMVERadbgDbziCRJru9n6tzEWrASVpEdrZrFidt1RDfn4yA3

Ensuite, nous utilisons la commande hd-public pour générer la séquence correspondante de deux clés publiques :

$ bx hd-public --index 0 &lt; compte  
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz5wzaSW4FiGrseNDR4LKqTy  
  
$ bx hd-public --index 1 &lt; compte  
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQDhohEcDFTEYMvYzwRD7Juf8

Les clés publiques peuvent également être dérivées de leurs clés privées correspondantes à l'aide de la commande hd-to-public :

$ bx hd-privé --index 0 &lt; compte | bx hd-to-public  
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz5wzaSW4FiGrseNDR4LKqTy  
  
$ bx hd-privé --index 1 &lt; compte | bx hd-to-public  
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQDhohEcDFTEYMvYzwRD7Juf8

Nous pouvons générer un nombre pratiquement illimité de clés dans une chaîne déterministe, toutes dérivées d'une seule graine. Cette technique est utilisée dans de nombreuses applications de portefeuille pour générer des clés qui peuvent être sauvegardées et restaurées avec une seule valeur de départ. C'est plus facile que d'avoir à sauvegarder le portefeuille avec toutes ses clés générées de manière aléatoire à chaque fois qu'une nouvelle clé est créée.

La graine peut être encodée à l'aide de la commande mnemonic-encode :

$ bx hd-mnemonic &lt; graine &gt; mots  
adorent répéter la pire vision, en particulier les pouces du voile, femme, rappel de la distribution, apprécient

La graine peut ensuite être décodée à l'aide de la commande mnemonic-decode :

$ bx décodage-mnémonique &lt; mots  
eb68ee9f3df6bd4441a9feadec179ff1

L'encodage mnémonique peut faciliter l'enregistrement et même la mémorisation de la graine.