

# CocosBCX - Traducción al español

Alexis Wolf Waisman

November 2019

## 1 Comenzando

### 1.1 Introduction

#### 1.1.1 ¿Qué es Cocos-BCX?

Cocos-BCX, la plataforma para la próxima generación de economía de juegos digitales, tiene como objetivo proporcionar a los desarrolladores de juegos una infraestructura de juegos de cadena de bloques fácil de usar y sólida, a la vez que ofrece a los usuarios un entorno de juego justo y abierto con datos transparentes y reglas claras. El mecanismo de consenso de la cadena de bloques subyacente está optimizado en base a la tecnología Graphene (grafeno). Con el sistema de contrato inteligente mejorado, Cocos-BCX es capaz de ofrecer un gran número de características para diversos juegos, tales como números aleatorios, sesión, temporizador y señales para verificación de calidad (HEARTHBEAT). Esperamos ayudar a los desarrolladores y jugadores a asegurar un beneficio consistente a través del modelo económico de activos digitales soportados por la tecnología de cadenas de bloques:

- Ayudar a los desarrolladores a asentar los contenidos que producen para obtener ganancias sostenibles a partir del uso, la gestión y la circulación de los activos de contenido proporcionando canales convenientes y descentralizados para la distribución de los juegos.
- Ayudar a los jugadores a convertir los datos generados por el tiempo y la energía que gastan, y los artículos obtenidos a través del consumo, en activos que puedan ser almacenados, gestionados y distribuidos de forma segura, otorgando a los jugadores los derechos para comercializarlos.

La Plataforma incluye:

1. Un marco de desarrollo que soporta múltiples sistemas operativos y cadenas de bloqueo.
2. Un IDE basado en datos para dApps que está totalmente programado y basado en componentes.

3. Un sistema de cadena de bloques y componentes funcionales esenciales para aplicaciones de alto rendimiento mejorados sobre la base del marco de la tecnología Graphene (grafeno).

Cocos-BCX habilita a los desarrolladores a programar, depurar y liberar dApps en conjunto con aplicaciones híbridas basadas en cadenas de bloques. Además, la plataforma integra un sistema de libro mayor distribuido (DLS) basado en una cadena de bloques, un sistema de billetera criptográfica y una plataforma de circulación de activos digitales, permitiendo a los activos internos que los mismos se almacenen fuera de la cadena de forma permanente y se puedan utilizar en modo multi-cadena. Teóricamente, el rendimiento de CocosChain TestNet puede alcanzar hasta 100.000 tps. Probamos 3.500 tps con intervalos de bloques de 3 segundos en un entorno experimental, es decir, se necesitan 3 segundos para que la información se transmita a toda la red.

El rendimiento mejorará aún más una vez que se haya completado el consenso definido en el contrato, el escalamiento multi-cadena y la delegación de testigos. Los algoritmos clave de la mayoría de los juegos pueden ser ejecutados en cadena, y la tecnología de "confirmación de transacciones con la latencia más baja" mejorará aún más la experiencia del proceso de circulación de activos. La billetera basada en CocosChain TestNet puede integrarse en la plataforma de circulación de activos; en donde los usuarios pueden evaluar el valor de las monedas, objetos y cuentas del juego en función del tipo de cambio de los activos digitales del juego con respecto a la moneda base de la cadena principal. Cocos-BCX está soportado por COCOS Creator, un editor de juegos, mediante el cual el juego desarrollado tradicionalmente puede ejecutarse en el entorno de tiempo de ejecución de la cadena de bloques Cocos-BCX.

### **1.1.2 Características técnicas y funciones generales**

Cocos-BCX apunta a construir un entorno de ejecución completo para juegos multisistema, proporcionando a los desarrolladores de juegos la máxima comodidad y un ecosistema mejorado, a la vez que aporta a los jugadores una experiencia de juego totalmente nueva, así como formas de juego que son diferentes de las anteriores. Los usuarios disfrutarán de una autonomía considerable en la disposición de los activos del juego y de un entorno de juego justo y abierto. Por lo tanto, Cocos-BCX está diseñado para ofrecer las siguientes características técnicas, que se describirán de manera enunciativa pero no taxativa:

1. Un entorno de tiempo de ejecución de juego multiplataforma con interfaces interoperables de cadena de bloques.
2. Un mecanismo de consenso mejorado basado en la prueba de la participación de los delegados (DPoS) y en la modalidad de testigos delegados.
3. Una TestNet (red de pruebas) con transmisión de datos mejorada y una máquina virtual de alto rendimiento.
4. Una pasarela de intercambio entre cadenas que admite activos digitales homogéneos y no homogéneos.

5. BCX-NHAS-1808 estándar de activos digitales no homogéneos.
6. Un sistema mejorado de permisos de los activos.
7. Contratos inteligentes ejecutables entre bloques.
8. Operación de transacción atómica (Todo-o-nada).
9. Apoyo a las tareas de consenso a nivel sintáctico.
10. Soporte para el mecanismo de transacciones delegadas.
11. Consenso a pequeña escala y números aleatorios.
12. Proceso aleatorio confiable en la cadena.
13. Soporte para ciclos mínimos de validación de transacciones dentro de la cadena.
14. Compatibilidad con contratos inteligentes para funciones clave del juego, como temporizador preciso, modo de espera y señalizador (heartbeat).
15. Mecanismo de verificación de transacciones para evitar que BP/desarrollador haga trampas.

Mientras tanto, Cocos-BCX proporciona funciones que incluyen, pero no se limitan a:

- Interfaces inter-operables de activos sin intermediarios (props).
- Ejemplos de plataformas de circulación de activos no homogéneos.
- Autonomía del jugador y el mecanismo Smithy.
- IDE visualizado (programa del juego y edición visual del contrato).
- Cartera criptográfica completa, sistema de cuentas de usuario y explorador de bloques.
- Sistema iterativo de contratos inteligentes.

### 1.1.3 Versiones

Componente	Versión
witness_node	
cli_wallet	
JS SDK	

#### 1.1.4 Requerimientos Técnicos

##### Programación LUA

Dado el uso extensivo del lenguaje Lua en el desarrollo de juegos tradicionales y su accesibilidad, nuestra máquina virtual de contractual actualmente soporta Lua (Lua 5.3) y cubrirá además otros lenguajes como JavaScript, Solidity entre otros; Es fácil comenzar con Lua leyendo el Manual de Referencia de Lua 5.3: <https://www.lua.org/manual/5.3/>

##### Desarrollo Front-End

Las aplicaciones de la cadena de bloques pueden distinguirse de las tradicionales por las dos fórmulas siguientes:

- **Aplicación = Frontend + Servidor**
- **DApp = Frontend + Contratos**

En los contratos proporcionados por Cocos-BCX de Lua, existen las siguientes 3 opciones para el front-end:

- Original Web JS  
Este es un método tradicional de desarrollo WEB, que consiste en llamar al SDK de JS proveído por Cocos-BCX para interactuar con el sistema Blockchain.
- Nativo  
Este se asimila al desarrollo en IOS y en Android que básicamente llama a la API nativa proveída por Cocos-BCX.
- Creador  
Este modelo es familiar para los desarrolladores de videojuegos. Se ha integrado al IDE de Cocos-BCX el creador; este hace fácil el despliegue de aplicaciones multiplataforma.

Líneas de comandos Las operaciones por líneas de Comandos son requeridas para el desarrollo de nodos y billeteras, por lo tanto, es muy recomendable tener conocimientos respectivos a los sistemas y consolas de Linux.

#### 1.1.5 Entornos de desarrollo integrado

- Creator
- VSCode
- Sublime Text
- Atom

## **1.2 Instalación**

### **1.2.1 Componentes**

nodo Testigo

programa nodo de la blockchain.

billetera cliente

Una billetera que soporta todas las instrucciones por líneas de comando cuya interacción se realiza con la blockchain. Los documentos técnicos específicos se encuentran en: <https://dev.cocosbcx.io/docs/22-cliwallet>

### **1.2.2 Componentes**

Compilación de código fuente Aún por desarrollar

### **1.2.3 Instalación desde los binarios**

Ambiente Ubuntu

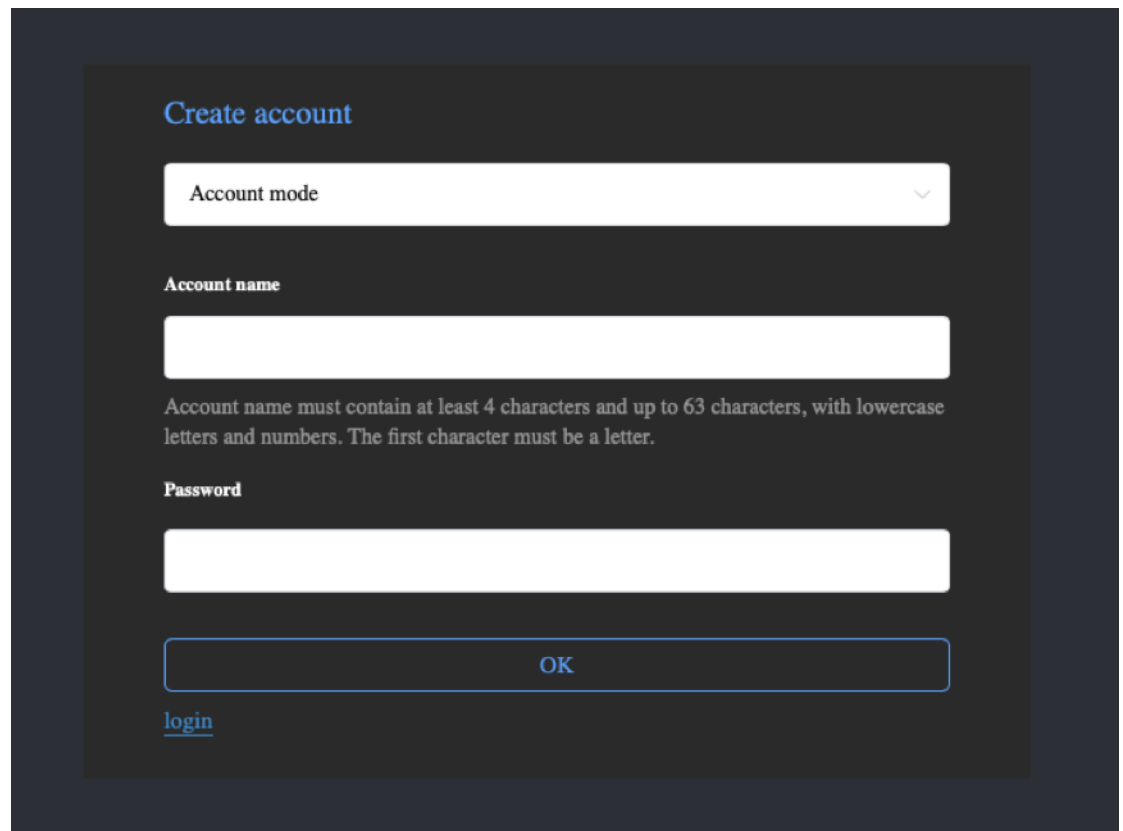
### **1.2.4 Instalación con Docker**

Aún por desarrollar

## **1.3 Crear una cuenta**

*Registro*

Te puedes registrar en la terminal de Cocos-BCX en <http://cocos-terminal.com/>. Todavía hay mucho para mejorar en cuanto a "experiencia de usuario" de la terminal, a pesar de ello, esta equipada con un montón de características; incluidas las funciones del núcleo de billetera y el browser.

A dark-themed 'Create account' form. At the top, the title 'Create account' is in blue. Below it is a white dropdown menu labeled 'Account mode' with a downward arrow. Next is a white input field for 'Account name'. Below the input field, a grey text block provides instructions: 'Account name must contain at least 4 characters and up to 63 characters, with lowercase letters and numbers. The first character must be a letter.' This is followed by a white input field for 'Password'. At the bottom of the form is a blue-outlined button labeled 'OK'. Below the button is a blue, underlined link labeled 'login'.

Create account

Account mode

Account name

Account name must contain at least 4 characters and up to 63 characters, with lowercase letters and numbers. The first character must be a letter.

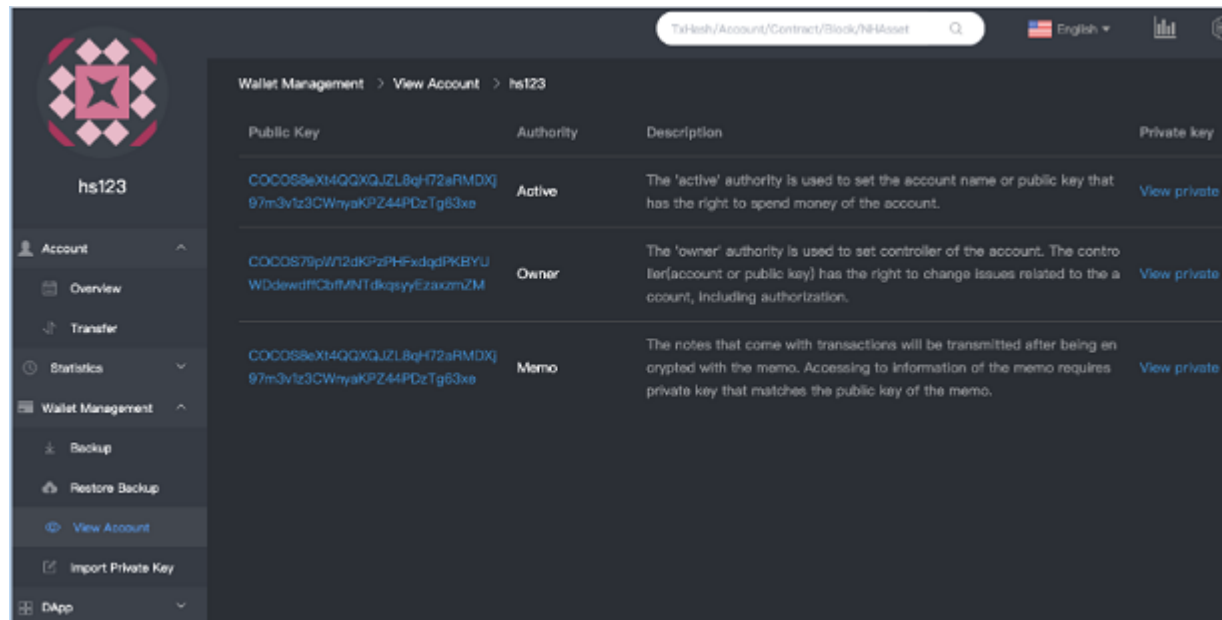
Password

OK

[login](#)

*Ver claves privadas*

Para ver tus claves privadas debes hacer clic en "ver cuenta" dentro de la barra lateral, luego hacer clic en "copiar clave privada".



## 1.4 transferencia

autenticándose en la billetera por CLI

1. conseguir el ejecutable de línea de comando. (Archivo: cli\_wallet) y copiarlo en el directorio destino.
2. Dirigirse al directorio donde la billetera CLI fué alojada. Ejecutar los siguientes comandos para ingresar a la misma.  
*comando: -chain-id [ID de cadena] [dirección del nodo RPC (testigo)] -r [ dirección desde la cual el servicio de RPC va a estar escuchando.]*

Listing 1: Autenticación

```
#!/bin/bash
./cli_wallet --chain-id
81003974d328ff17b64076928ab87b24d7dffbc87df3d4cde89d2fa1877e4f6a -s
ws://127.0.0.1:8070 -r 127.0.0.1:8099
# todo en una linea !!!!
```

```

Logging RPC to file: logs/rpc/rpc.log
3552248ms th_a      main.cpp:131      main      ] key_to
_wif( committee_private_key ): 5KCBDTcyDqzsqehcb52tW5nU6pXife6V2rX9Yf7c3saYSzbDZ
5W
3552249ms th_a      main.cpp:135      main      ] nico_p
ub_key: COCOS7yE9skpBAirth3eSNMRtwqljYswEE3uSbbuAtXTz88HtbpQsZf
3552249ms th_a      main.cpp:136      main      ] key_to
_wif( nico_private_key ): 5KAUeN3Yv5lFzpLGGf4S1ByKpMqVFNzXTJK7euqc3NnaaLzlGJm
Starting a new wallet with chain ID 9fc429a48b47447afa5e6618fde46dla5f7b2266f00c
e60866f9fdd92236e137 (from CLI)
3552250ms th_a      main.cpp:183      main      ] wdata.
ws_server: ws://47.93.62.96:8020
3552266ms th_a      main.cpp:188      main      ] wdata.
ws_user:  wdata.ws_password:
Please use the set_password method to initialize a new wallet before continuing
3552321ms th_a      main.cpp:227      main      ] Listen
ing for incoming RPC requests on 127.0.0.1:8099

```

Configurar clave para la billetera

1. La primera vez que se realiza la autenticación se debe establecer una contraseña. *comando: set\_password*

Listing 2: Configurar password

```
set_password xxxx
```

```

new >>> set_password
set_password
null
locked >>>

```

2. Después de configurar la clave, se necesita desbloquear la billetera.

Listing 3: Configurar password

```
unlock xxxx
```

Importar cuenta

1. autenticarse en la billetera y ejecutar los siguientes comandos para importar la cuenta.  
*comando: import\_key [nombre de usuario] [claves privadas de usuario]*  
*clave privada: para ver la misma por favor remítase a la sección/subsección 1.3 de este mismo documento, en el apartado: **ver claves privadas***



Listing 4: importar claves privadas

```
import_key official -account 5KaVpJa9G4oqA5WHcSGitauFRuzdHcPVEAaESaA
```

```
unlocked >>> import_key chandlerette ["5KaVpJa9G4oqA5WHcSGitauFRuzdHcPVEAaESaA"] true
import_key chandlerette ["5KaVpJa9G4oqA5WHcSGitauFRuzdHcPVEAaESaA"] true
2236777ms th_a wallet.cpp:869 save_wallet_file ] saving
wallet to file wallet.json
2236779ms th_a wallet.cpp:542 copy_wallet_file ] backing
up wallet wallet.json to after-import-key-f9ee9ead.wallet
true
```

Importar los activos a la billetera

1. autenticarse en la billetera y ejecutar los siguientes comandos para importar la cuenta.

*comando: import\_balance [nombre de usuario] [clave privada correspondiente a la dirección del activo] [true/false]*

*contexto: una nueva cadena cuyos activos iniciales no fueron exportados*

Listing 5: importar claves privadas

```
import_balance official -account
["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
```

```
unlocked >>> import_balance chandlerette ["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
import_balance chandlerette ["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
2476253ms th_a wallet.cpp:4885 import_balance ] balances: []
[]
```

Importar los activos a la billetera

1. autenticarse en la billetera y ejecutar los siguientes comandos para importar la cuenta.

*comando: import\_balance [nombre de usuario] [clave privada correspondiente a la dirección del activo] [true/false]*

*contexto: una nueva cadena cuyos activos iniciales no fueron exportados*

Listing 6: importar claves privadas

```
import_balance official -account
["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
```

```
unlocked >>> import_balance chandlerette ["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
import_balance chandlerette ["5KAUeN3Yv51FzpLGGf4S1ByKpMqVFNzXTJK7euqc3L"] true
2476253ms th_a wallet.cpp:4885 import_balance ] balances: []
[]
```