



CESS Network

The Decentralized Data Infrastructure

Episode 8: DApp Development using Ink! Smart Contract



<https://www.cess.network>



Course Logistics

Course Website: <https://course.cess.network/>

- Episode 1** - . - . - . - . - . - . CESS Network Introduction
- Episode 2** - . - . - . - . - . - . CESS Architecture & Key Technologies
- Episode 3** - . - . - . - . - . - . CESS Ecosystem and Applications
- Episode 4** - . - . - . - . - . - . CESS Nodes & CESS Account Setup
- Episode 5** - . - . - . - . - . - . Demo: Running a Consensus Node
- Episode 6** - . - . - . - . - . - . Demo: Running a Storage Node
- Episode 7** - . - . - . - . - . - . CESS DeOSS and DeOSS REST API
- Episode 8** - . - . - . - . - . - . dApp Development using ink! Smart Contract
- Episode 9** - . - . - . - . - . - . dApp Development using Solidity Smart Contract
- Episode 10** - . - . - . - . - . - . Building Custom Pallet



Why Choose Ink! Smart Contract?



Access to Polkadot Ecosystem

CESS developers can leverage the extensive network and resources of the Polkadot ecosystem.

Enhanced Security

Developers benefit from the security features of Rust, minimizing the risk of smart contract vulnerabilities.

Performance and Efficiency

Ensures high performance and efficient execution of smart contracts on CESS

WebAssembly (WASM) Compatibility

Ink! Compiles to WASM, which is supported by various blockchains, allowing cross-chain deployment

Developer-Friendly Environment

With robust tooling, documentation and support Ink! Attracts wide pool of developers.

Advanced Features

Offers upgradable contracts and modular design for flexible and maintainable DApps on CESS

Ink! Smart Contract Dev Prerequisites



- Install Rust & Cargo



```
curl https://sh.rustup.rs -sSf | sh
```

- Install Ink! CLI



```
rustup component add rust-src  
cargo install --force --locked cargo-contract
```

Flipper Smart Contract Walkthrough



A simple Flipper smart contract that flips a boolean value from true to false and vice versa

- Create a Smart Contract



```
cargo contract new flipper
cd flipper
```



```
flipper
├─ lib.rs          <-- Contract Source Code
├─ Cargo.toml      <-- Rust Dependencies and ink! Configuration
└─ .gitignore
```

Lib.rs Structure



```
#![cfg_attr(not(feature = "std"), no_std, no_main)]

#[ink::contract]
pub mod flipper {
    ...
}
```



```
#[ink(storage)]
pub struct Flipper {
    value: bool,
}
```

Macros

cfg_attr: Rust attribute that conditionally configures attributes indicating the project to be *no_std* or no standard library with *no_main* for non-main function execution.

ink::contract: Marks the beginning of the Ink smart contract module making flipper module as the ink smart contract

ink(storage): Specifies the storage layout of the contract.

Flipper Implementation



```
impl Flipper {  
  /// Creates a new flipper smart contract initialized with the given value.  
  #[ink(constructor)]  
  pub fn new(init_value: bool) -> Self {  
    Self { value: init_value }  
  }  
  
  /// Creates a new flipper smart contract initialized to `false`.  
  #[ink(constructor)]  
  pub fn new_default() -> Self {  
    Self::new(Default::default())  
  }  
  
  /// Flips the current value of the Flipper's boolean.  
  #[ink(message)]  
  pub fn flip(&mut self) {  
    self.value = !self.value;  
  }  
  
  /// Returns the current value of the Flipper's boolean.  
  #[ink(message)]  
  pub fn get(&self) -> bool {  
    self.value  
  }  
}
```

Flipper Implementation

Contains functions that are specific to Flipper contract.

`new` and `new_default` are constructors that initialize the contract marked with `#[ink(constructor)]`

`#[ink(message)]`: defines the external interface for the smart contract which represents an action that external entities can invoke.

- `flip()`: Flips the boolean value
- `get()`: Retrieves the current boolean value

Test Module



```
#[cfg(test)]
mod tests {
    use super::*;

    #[ink::test]
    fn default_works() {
        let flipper = Flipper::default();
        assert_eq!(flipper.get(), false);
    }

    #[ink::test]
    fn it_works() {
        let mut flipper = Flipper::new(false);
        assert_eq!(flipper.get(), false);
        flipper.flip();
        assert_eq!(flipper.get(), true);
    }
}
```

`#[ink(test)]`: Macro indicates that tests are only compiled when running tests.

Verifies the functionality of Flipper contract.

E2E(End to End) Test



```
mod e2e_tests {
  use super::*;
  use ink_e2e::build_message;

  type E2EResult<T> = std::result::Result<T, Box<dyn std::error::Error>>;

  #[ink_e2e::test]
  async fn it_works(mut client: ink_e2e::Client<C, E>) -> E2EResult<()> {
    // Test logic goes here...
    Ok(())
  }

  #[ink_e2e::test]
  async fn default_works(mut client: ink_e2e::Client<C, E>) -> E2EResult<()> {
    Ok(())
  }
}
```

#[ink_e2e::test]: Indicates that the test are only compiled when both standard test and specific E2E test features are enabled

Uses Ink! E2e crate to conduct end-to-end testing simulating interactions with the Flipper contract in a controlled environment



Demo

Deploying **Ink!** Smart Contract on CESS



Deploying Ink! Smart Contract on CESS



Step 1: Clone and Build CESS source code

```
git clone https://github.com/CESSProject/cess.git --branch v0.7.4
```

```
cd cess
```

```
cargo build
```

Step 2: Start CESS node in Dev Mode

```
target/debug/cess-node --dev
```

```
2023-08-23 18:01:10 CESS Node
2023-08-23 18:01:10 🙌 version 0.6.0-unknown
2023-08-23 18:01:10 ❤️ by CESS LAB, 2017-2023
2023-08-23 18:01:10 📄 Chain specification: Development
2023-08-23 18:01:10 🏠 Node name: alike-stretch-9767
2023-08-23 18:01:10 👤 Role: AUTHORITY
2023-08-23 18:01:10 🗄 Database: RocksDb at /tmp/substrateZdfnHg/chains/dev/db/full
2023-08-23 18:01:10 🐘 Native runtime: cess-node-109 (cess-node-1.tx1.au1)
channel 17: open failed: connect failed: Connection refused
2023-08-23 18:01:13 [0] 🐘 generated 1 npos voters, 1 from validators and 0 nominators
2023-08-23 18:01:13 [0] 🐘 generated 1 npos targets
2023-08-23 18:01:15 ↙ Initializing Genesis block/state (state: 0xd45c...c2b1, header-hash: 0x126c...4ad9)
2023-08-23 18:01:15 😊 Loading GRANDPA authority set from genesis on what appears to be first startup.
channel 17: open failed: connect failed: Connection refused
2023-08-23 18:01:17 😊 Creating empty RRSC epoch changes on what appears to be first startup.
2023-08-23 18:01:17 Using default protocol ID "sup" because none is configured in the chain specs
2023-08-23 18:01:17 🏠 Local node identity is: 12D3KooWPaspHpeUHLxY4iRhBL3cbMEscfhAFTuSEgPLWaJFpdgt
```

Deploying Ink! Smart Contract on CESS



Step 3: [Open Substrate Contracts UI](#)

<https://contracts-ui.substrate.io/?rpc=ws://127.0.0.1:9944>

A screenshot of the Substrate Contracts UI interface. The interface is divided into three main sections. On the left is a sidebar with a 'Local Node' dropdown, buttons for 'Add New Contract' and 'All Contracts', and a 'Your Contracts' section showing 'None yet' and an 'Upload one' link. The central area is titled 'Contracts' and contains a message: 'You haven't uploaded any contracts yet on this browser.' with a folder icon and a green link 'Upload a new contract'. On the right is a 'Tell us more about yourself' section with a survey button, and a table showing chain information.

Chain Name	Chain Type
CESS Node	Development
Highest Block	Token
#1,185	Unit

Step 4: Click on Upload a new Contract

Upload *target/ink/flipper.contract* file


Deploying Ink! Smart Contract on CESS



Step 5:

Give a name to your Contract in “Contract Name” field and Click “Next”


Account ⓘ

 **alice**
5Grw...utQY

Contract Name ⓘ

Flipper Contract

Upload Contract Bundle ⓘ

 flipper.contract (27.00kb) ×

Valid contract bundle!

Metadata

Contract Hash

0x3cb08c19c86ce5b75ed720b02605d0e5770bd7d6a166ecd2e32d66dc5667b6d1

Language

ink! 4.2.1

Contract version

0.1.0

Compiler

rustc 1.71.1

Authors

[your_name] <[your_email]>

▼ new(initValue: bool)

▼ default()

▼ flip(): Result<Null, InkPrimitivesLangError> 🚩

▼ get(): Result<bool, InkPrimitivesLangError>

Next

Deploying Ink! Smart Contract on CESS



Step 6:

Instantiate Contract with
Default Values

Upload and Instantiate Contract

You can instantiate a new contract from an existing code bundle [here](#).

Deployment Constructor ⓘ

`new(initValue: bool)`

initValue: bool

false

Deployment Salt ⓘ

0x027cf3f133a60f69ff8729b00cd96d7b853cc9708b571697c7de8af976717039

☒

RefTime Limit ⓘ

327223355

Using Estimation · [Use Custom](#)

ProofSize Limit ⓘ

100

Using Estimation · [Use Custom](#)

Storage Deposit Limit ⓘ

Do not use

☐

Go Back

Next

Deploying Ink! Smart Contract on CESS




Congratulations!


You have Successfully
Instantiated Smart
Contract

Upload and Instantiate Contract

You can instantiate a new contract from an existing code bundle [here](#).

 `instantiateWithCode`
queued

Account

 **alice**
5Grw...utQY

Name

Flipper Contract

Weight

327,223,355

Data

0x0061...c80a01

Code Hash

0xde62...771d7b

[Go Back](#)

[Upload and Instantiate](#)

Interacting With Ink! Smart Contract



- Select caller account
- Select function to execute
- Enter the desired values
- Call Contract

Flipper Contract


⌂ Reinstantiate 🗑️

You instantiated this contract `cXj1...Xnz3` from `flipper` on 24 Aug

📄 Metadata

⚙️ Interact

Caller ⓘ

 **alice**
5Grw...utQY

▼

Message to Send ⓘ

`flip(): Result<Null, InkPrimitivesLangError>` 🗑️

▼

RefTime Limit ⓘ

Using Estimation · [Use Custom](#)

ProofSize Limit ⓘ

Using Estimation · [Use Custom](#)

Storage Deposit Limit ⓘ

☒

Call contract

Dry-run outcome

Contract call will be successful!

Execution result

Ok 🗑️

GasConsumed

refTime: 592753250 🗑️

proofSize: 13593 🗑️

GasRequired

refTime: 4006871040 🗑️

proofSize: 131072 🗑️

StorageDeposit

charge: 0 🗑️

Transactions log

No transactions yet.



Demo

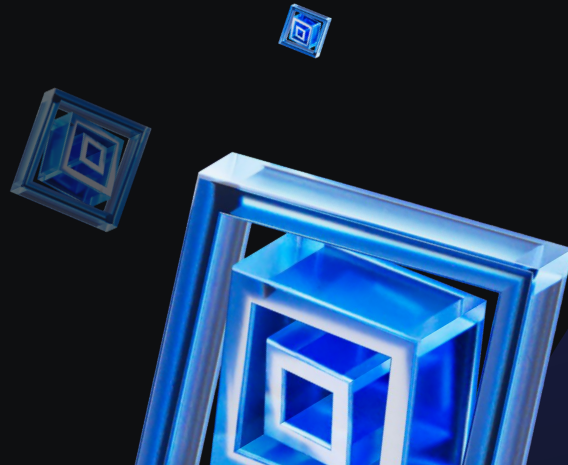
Interacting with Ink! Smart Contract Using
The useink Library





Demo

NFT Marketplace Example





Thank you for watching

Please Join Our Community





CESS Network - Episode 8

DApp Development using Ink!
Smart Contract



<https://www.cess.network>

