SMART BEAVER

ETHWarsaw23

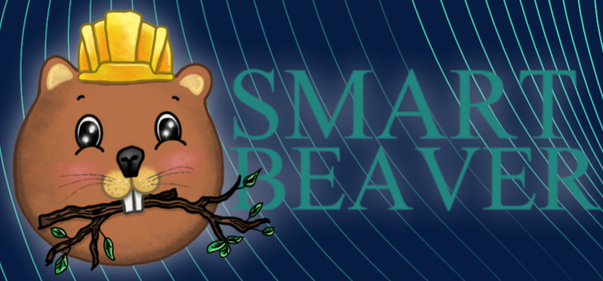
Are you tired of using OpenBrush? Are you frustrated with their custom library, lack of updates, and limited features?

Do you dream of having a wizard that can easily configure and deploy contracts written in **clean ink**!?

SMART BEAVER

Welcome to Smart Beaver! We specialize in building contracts as **sturdy as dams**.

3 Simple steps



Configure

Choose the smart contract standard you want to deploy, and then configure its properties

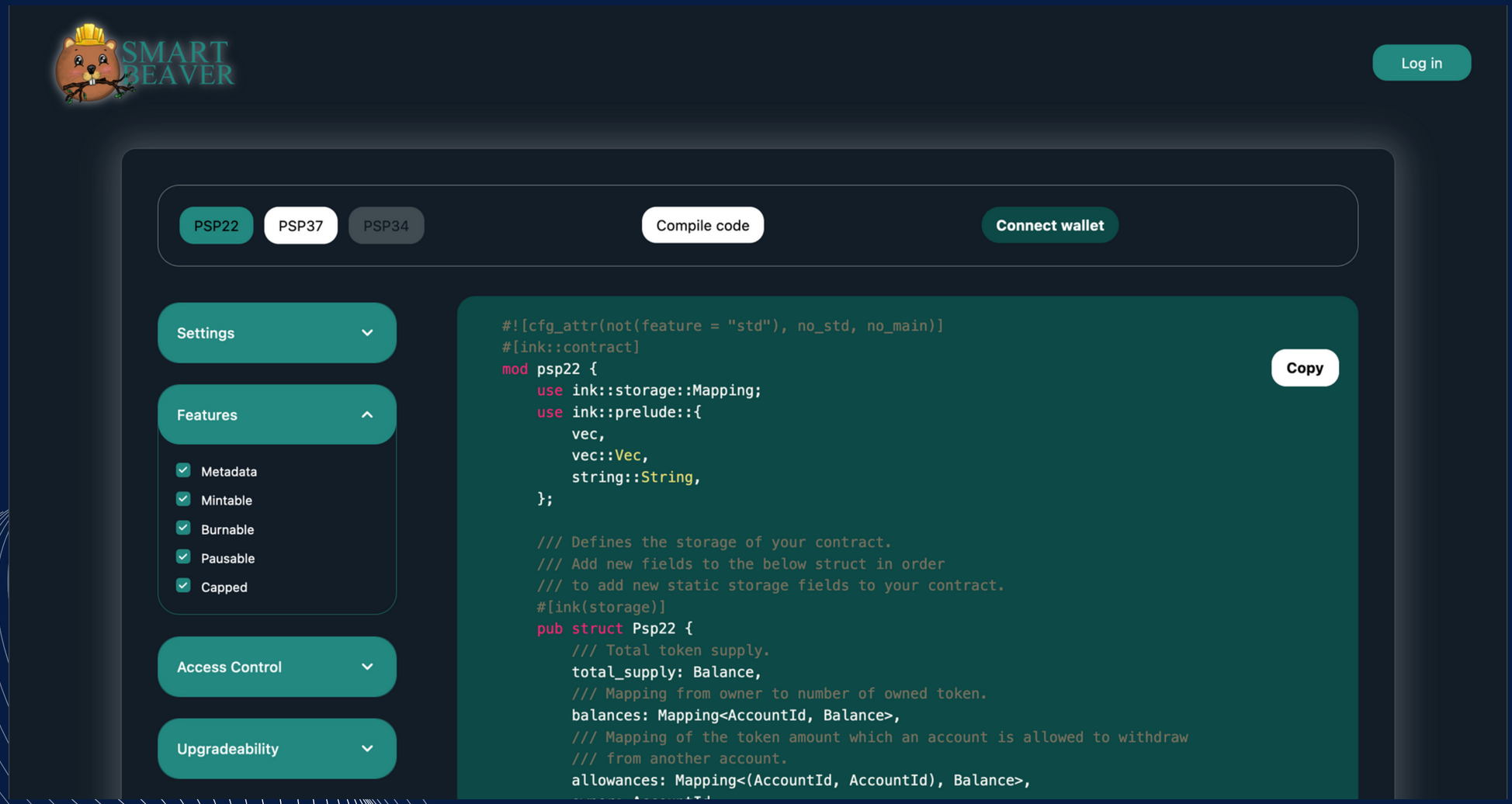
Inspect

Inspect the generated source code; you can choose to deploy it or use it as a template.

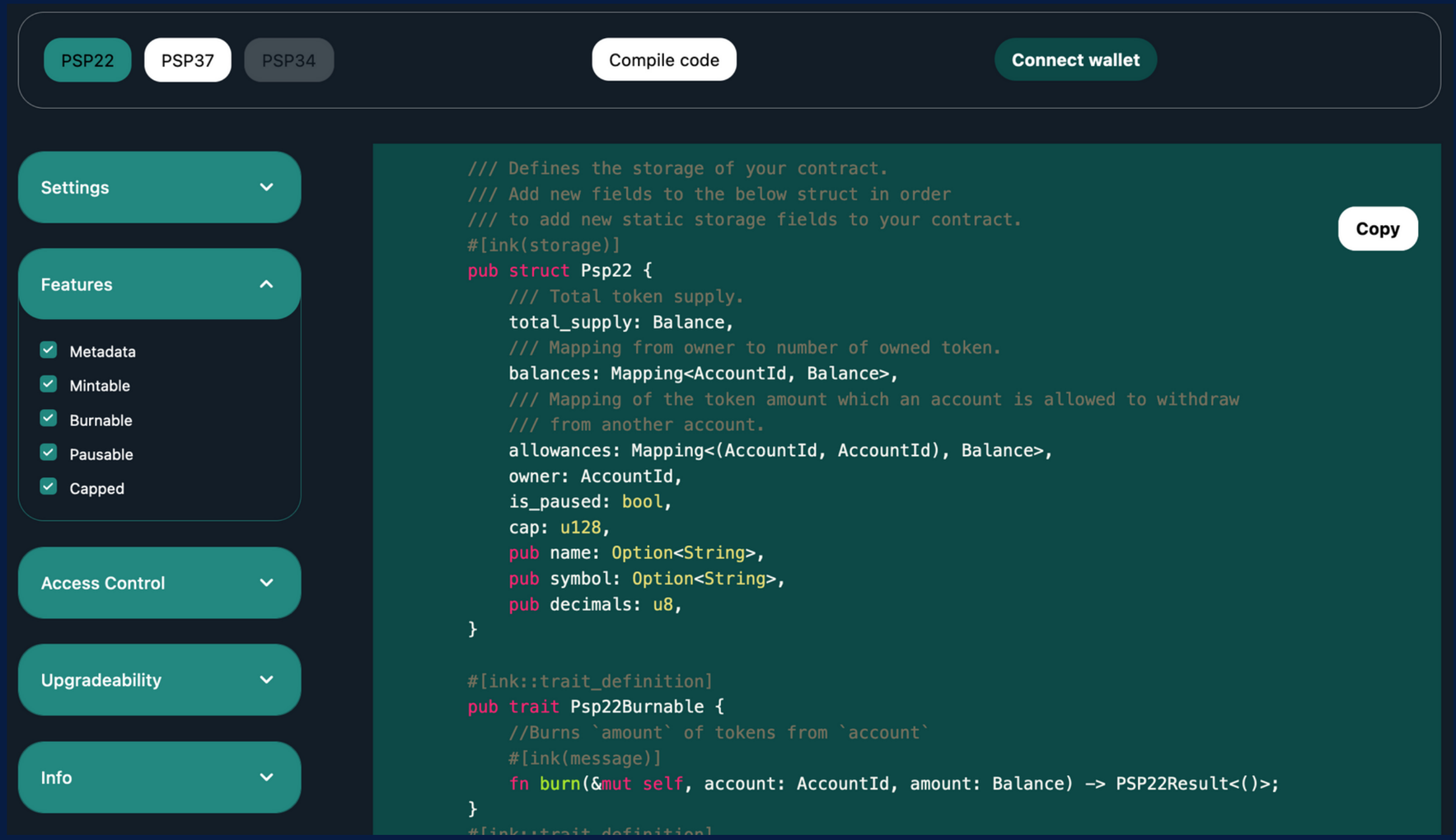
Deploy

Connect your wallet and either deploy the contract or download the compiled WASM.

Configure

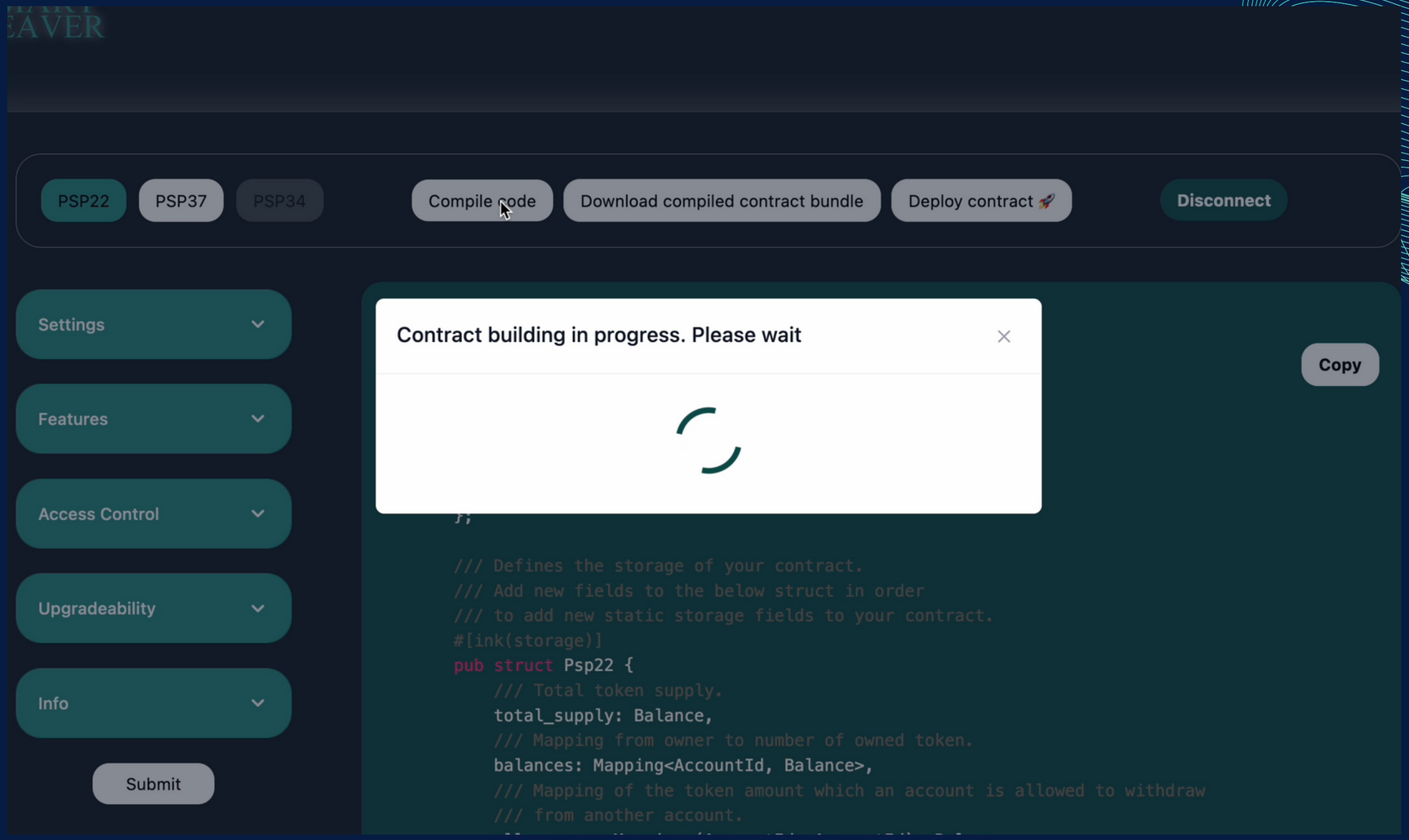

SMART BEAVER
Log in
PSP22
PSP37
PSP34
Compile code
Connect wallet
Settings
Features
Metadata
Mintable
Burnable
Pausable
Capped
Access Control
Upgradeability
Copy
#![cfg_attr(not(feature = "std"), no_std, no_main)]
#[ink::contract]
mod psp22 {
use ink::storage::Mapping;
use ink::prelude::{
vec,
vec::Vec,
string::String,
};
/// Defines the storage of your contract.
/// Add new fields to the below struct in order
/// to add new static storage fields to your contract.
#[ink(storage)]
pub struct Psp22 {
/// Total token supply.
total_supply: Balance,
/// Mapping from owner to number of owned token.
balances: Mapping<AccountId, Balance>,
/// Mapping of the token amount which an account is allowed to withdraw
/// from another account.
allowances: Mapping<(AccountId, AccountId), Balance>,

Inspect

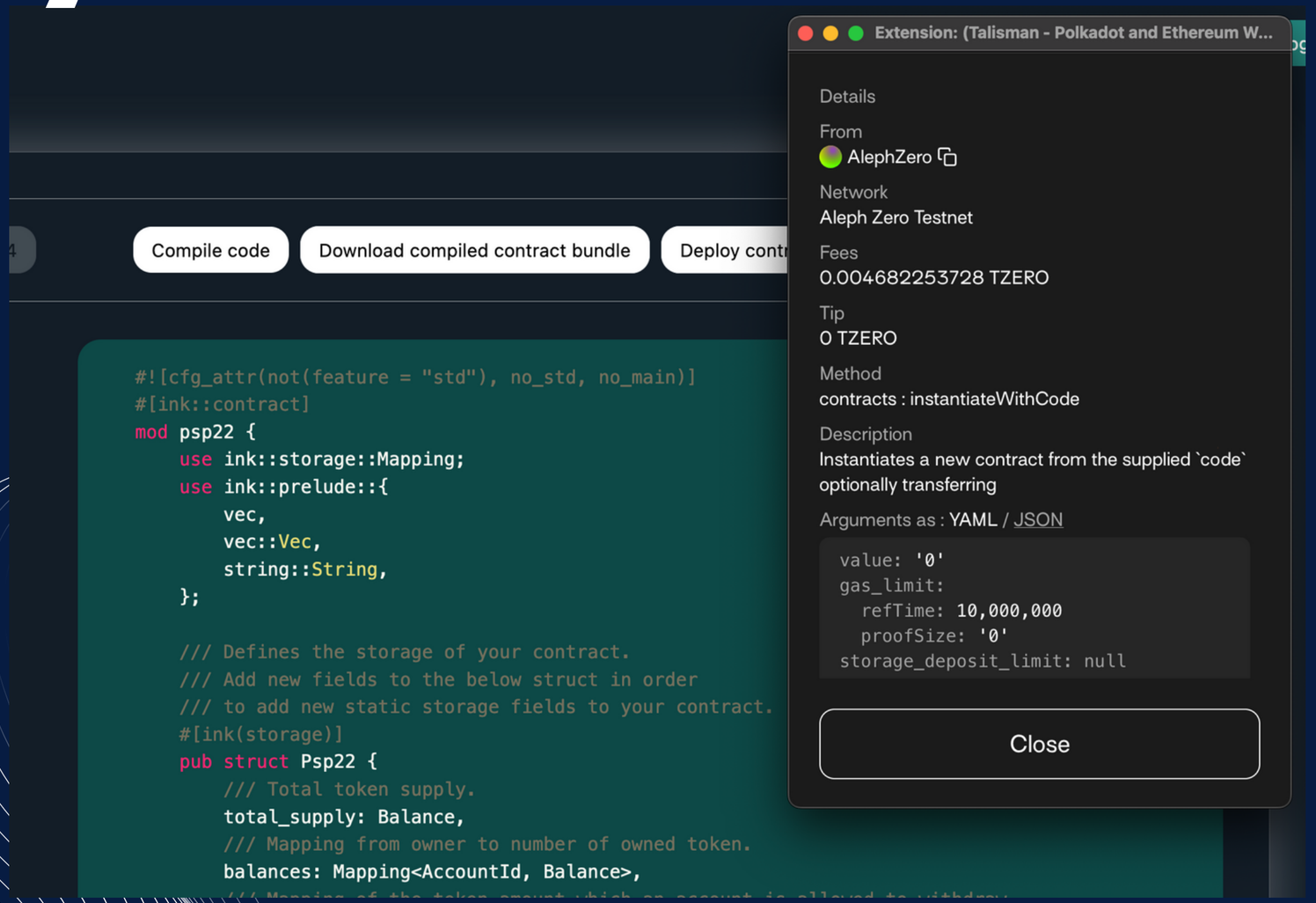


Contracts in clean ink!

Compile


PSP22
PSP37
PSP34
Compile code
Download compiled contract bundle
Deploy contract
Disconnect
Settings
Features
Access Control
Upgradeability
Info
Submit
Contract building in progress. Please wait
Copy
/// Defines the storage of your contract.
/// Add new fields to the below struct in order
/// to add new static storage fields to your contract.
#[ink(storage)]
pub struct Psp22 {
/// Total token supply.
total_supply: Balance,
/// Mapping from owner to number of owned token.
balances: Mapping<AccountId, Balance>,
/// Mapping of the token amount which an account is allowed to withdraw
/// from another account.

Deploy


Compile code
Download compiled contract bundle
Deploy contr
#![cfg_attr(not(feature = "std"), no_std, no_main)]
#[ink::contract]
mod psp22 {
use ink::storage::Mapping;
use ink::prelude::{
vec,
vec::Vec,
string::String,
};
/// Defines the storage of your contract.
/// Add new fields to the below struct in order
/// to add new static storage fields to your contract.
#[ink(storage)]
pub struct Psp22 {
/// Total token supply.
total_supply: Balance,
/// Mapping from owner to number of owned token.
balances: Mapping<AccountId, Balance>,
Extension: (Talisman - Polkadot and Ethereum W...
Details
From
AlephZero
Network
Aleph Zero Testnet
Fees
0.004682253728 TZERO
Tip
0 TZERO
Method
contracts : instantiateWithCode
Description
Instantiates a new contract from the supplied `code` optionally transferring
Arguments as : YAML / JSON
value: '0'
gas_limit:
refTime: 10,000,000
proofSize: '0'
storage_deposit_limit: null
Close

Currently supported

1 STANDARD

5 EXTENSIONS

3 SETTINGS

4
Standars

All
Extensions

Docs
with
examples

Our future

SMART BEAVER

Smart Beaver

Izabela Łaszczuk

Piotr Swierzy

Maciej Malik