CESS Network

The Decentralized Data Infrastructure

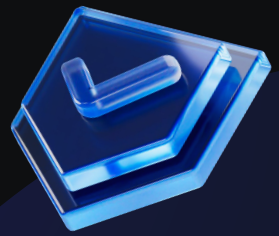
Episode 10: Building Custom Pallets

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 build your own pallets? Purpose.

CESS

Pros:

- Flexible and the most powerful
- Can alter consensus mechanism, transaction fee, block time, etc.

Cons:

- A high technical barrier
  - learning Rust and Substrate macro
  - Familiarize with existing codebase
- Running your own chain requires more devOps and evening maintaining validator community.

Purpose

- Tailored Features: Developers can create features specifically designed to meet the needs of CESS users, enhancing the network's functionality and user experience.
- Expand Network Capabilities: Custom pallets can introduce features that drive the adoption and growth of the CESS network, attracting more users, developers, and stakeholders.
- Addressing Unique Challenges: Custom pallets allow developers to address specific challenges faced by the CESS network, providing solutions that are tailored to its unique requirements.

How to Contribute to CESS with Custom Pallets

CESS

Understand the CESS Codebase

- Start by becoming familiar with CESS's architecture and existing codebase.

Identify Improvement Areas

- Look for opportunities to add new functionalities or enhance existing ones.

Develop Custom Pallets

- Create custom pallets tailored to the unique needs and challenges of the CESS network.

Engage with the Community

- Submit your custom pallets for review and participate in community discussions.

Participate in Governance

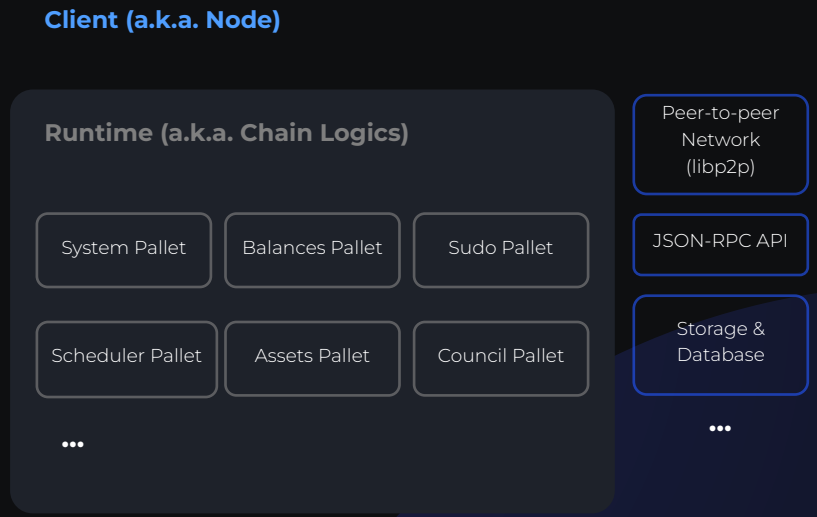
- Get involved in CESS's decentralized governance to influence protocol updates and policies.

Contribute to Ecosystem Growth

- Help shape the future of CESS by developing innovative solutions and enhancing the network’s capabilities.

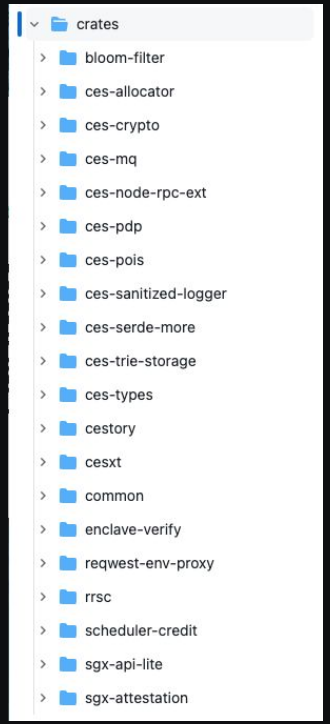
High Level Description of CESS (Substrate) Framework

- **Client:** p2p networking, low-level library inclusion of (merkle tree) storage & database, JSON-RPC API
- **Runtime:** a composition of pallets connected together to form the core chain logics and expose the extrinsics that are publicly available.
- **Pallets:** modules that perform a specific set of functions on-chain.

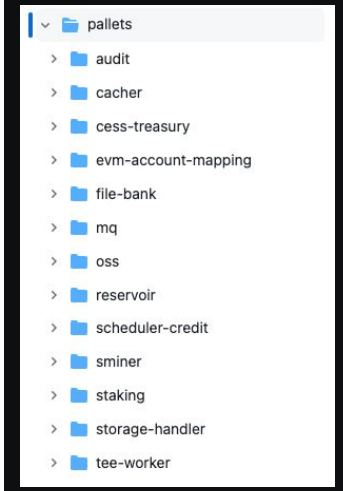


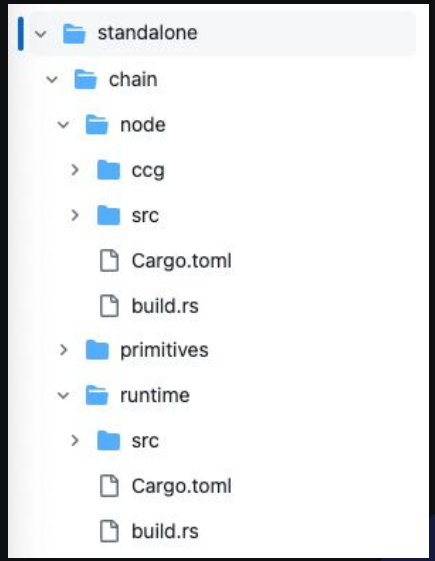
CESS Source Code

src: https://github.com/CESSProject/cess/blob/main



Crates and Pallets (Libararies)

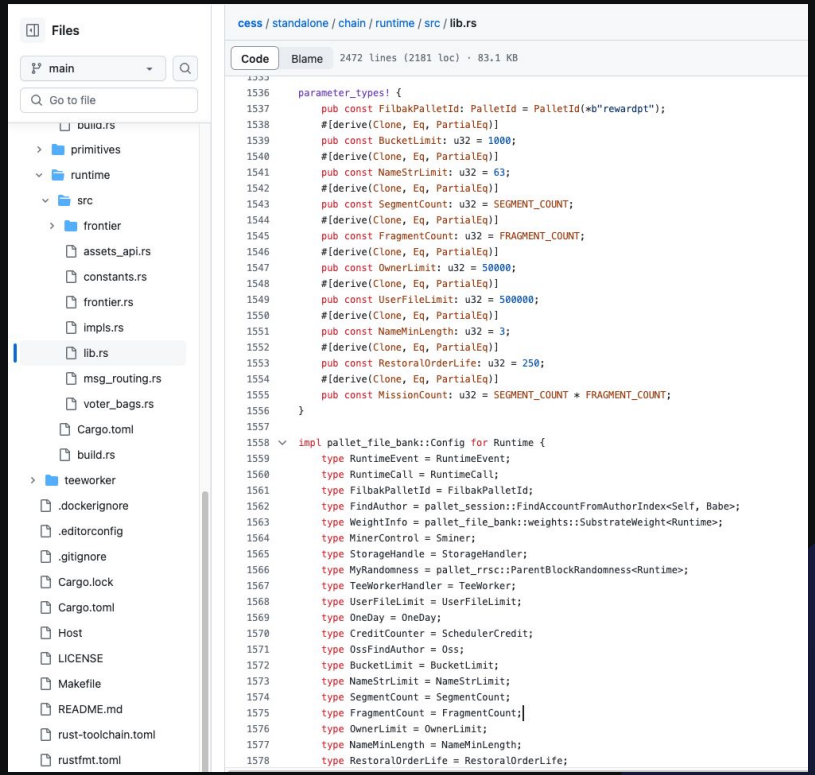




Node and Runtime

Overview of the CESS Node Runtime

Runtime implements the pallet : : Config



Structure of the Runtime

Put the pallet inside the construct_runtime!{...} macro

```
// Create the runtime by composing the FRAME pallets that were previously configured.
construct_runtime!(
    pub enum Runtime
    {
        // Basic stuff
        System: frame_system = 0,
        RandomnessCollectiveFlip: pallet_insecure_randomness_collective_flip = 1,
        Timestamp: pallet_timestamp = 2,
        Sudo: pallet_sudo = 3,
        Scheduler: pallet_scheduler = 4,
        Preimage: pallet_preimage = 5,
        Mmr: pallet_mmr = 6,

        // Account lookup
        // ...

        // Tokens & Fees
        Balances: pallet_balances = 10,
        TransactionPayment: pallet_transaction_payment = 11,
        Assets: pallet_assets = 12,
        AssetTxPayment: pallet_asset_tx_payment = 13,

        // Consensus
        // ...

        // Governance
        // ...

        // Smart contracts
        Contracts: pallet_contracts = 50,
        Ethereum: pallet_ethereum = 51,
        EVM: pallet_evm = 52,
        EVMChainId: pallet_evm_chain_id = 53,
        DynamicFee: pallet_dynamic_fee = 54,
        BaseFee: pallet_base_fee = 55,

        // CESS pallets
        // ...
        FileBank: pallet_file_bank = 60,
    }
);
```

Example: EVM Runtime Pallet

Even the EVM capability in CESS node is setup the same way.

```
parameter_types! {
  pub BlockGasLimit: U256 = U256::from(BLOCK_GAS_LIMIT);
  pub const GasLimitPovSizeRatio: u64 = BLOCK_GAS_LIMIT.saturating_div(MAX_POV_SIZE);
  pub PrecompilesValue: FrontierPrecompiles<Runtime> = FrontierPrecompiles::<_>::new();
  pub WeightPerGas: Weight = Weight::from_parts(weight_per_gas(BLOCK_GAS_LIMIT, NORMAL_DIS
}

impl pallet_evm::Config for Runtime {
  type FeeCalculator = BaseFee;
  type GasWeightMapping = pallet_evm::FixedGasWeightMapping<Self>;
  type WeightPerGas = WeightPerGas;
  type BlockHashMapping = pallet_ethereum::EthereumBlockHashMapping<Self>;
  type CallOrigin = EnsureAddressTruncated;
  type WithdrawOrigin = EnsureAddressTruncated;
  type AddressMapping = HashedAddressMapping<BlakeTwo256>;
  type Currency = Balances;
  type RuntimeEvent = RuntimeEvent;
  type PrecompilesType = FrontierPrecompiles<Self>;
  type PrecompilesValue = PrecompilesValue;
  type ChainId = EVMChainId;
  type BlockGasLimit = BlockGasLimit;
  type Runner = pallet_evm::runner::stack::Runner<Self>;
  type OnChargeTransaction = ();
  type OnCreate = ();
  type FindAuthor = FindAuthorTruncated<Babe>;
  type GasLimitPovSizeRatio = GasLimitPovSizeRatio;
  type Timestamp = Timestamp;
  type WeightInfo = pallet_evm::weights::SubstrateWeight<Self>;
}
```

CESS

Walkthrough

CESS Pallets

Pallets Integrated in CESS Node

CESS

Foundational

System frame, Timestamp pallet, Sudo pallet, Scheduler pallet, etc...

Governance

Council, TechnicalCommittee, Treasury, Bounties, Multisig, etc...

Account & Fee

Proxy pallet, Indices pallet, Balances pallet, Assets pallet, etc...

Smart Contract

Contracts (ink!), Ethereum, EVM, EVMChainId, DynamicFee, etc...

Consensus

Authorship pallet, Babe pallet, Grandpa pallet, Staking pallet, etc...

CESS Specific

FileBank, TeeWorker, Audit, Sminer, StorageHandler, etc...

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[frame_support::pallet]
pub mod pallet {
    // ...

    #[pallet::config]
    pub trait Config: frame_system::Config + sp_std::fmt::Debug {
        /// The overarching event type.
        type RuntimeEvent: From<Event<Self>> + IsType<<Self as frame_system::Config>::RuntimeEvent>;

        type WeightInfo: WeightInfo;

        type RuntimeCall: From<Call<Self>>;

        type FScheduler: ScheduleNamed<BlockNumberFor<Self>, Self::SProposal, Self::SPalletsOrigin>;

        type AScheduler: ScheduleAnon<BlockNumberFor<Self>, Self::SProposal, Self::SPalletsOrigin>;
        /// Overarching type of all pallets origins.
        type SPalletsOrigin: From<frame_system::RawOrigin<Self::AccountId>>;
        // ...

        /// pallet address.
        #[pallet::constant]
        type FilbakPalletId: Get<PalletId>;

        #[pallet::constant]
        type UserFileLimit: Get<u32> + Clone + Eq + PartialEq;

        #[pallet::constant]
        type OneDay: Get<BlockNumberFor<Self>>;

        // User defined name length limit
        #[pallet::constant]
        type NameStrLimit: Get<u32> + Clone + Eq + PartialEq;
        // Maximum number of containers that users can create.
        #[pallet::constant]
        type BucketLimit: Get<u32> + Clone + Eq + PartialEq;

        // ...
    }
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::event]
#[pallet::generate_deposit(pub(super) fn deposit_event)]
pub enum Event<T: Config> {
    //file upload declaration
    UploadDeclaration { operator: AccountOf<T>, owner: AccountOf<T>, deal_hash: Hash },
    //file uploaded.
    TransferReport { acc: AccountOf<T>, deal_hash: Hash },
    //File deletion event
    DeleteFile { operator:AccountOf<T>, owner: AccountOf<T>, file_hash: Hash },
    // ...
}
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::error]
pub enum Error<T> {
    Existed,

    FileExistent,
    //file doesn't exist.
    FileNonExistent,
    //overflow.
    Overflow,

    NotOwner,

    NotQualified,
    //It is not an error message for scheduling operation
    ScheduleNonExistent,
    // ...
}
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::storage]
#[pallet::getter(fn deal_map)]
pub(super) type DealMap<T: Config> = StorageMap<_, Blake2_128Concat, Hash, DealInfo<T>>;

#[pallet::storage]
#[pallet::getter(fn file)]
pub(super) type File<T: Config> =
    StorageMap<_, Blake2_128Concat, Hash, FileInfo<T>>;

#[pallet::storage]
#[pallet::getter(fn user_hold_file_list)]
pub(super) type UserHoldFileList<T: Config> = StorageMap<
    _,
    Blake2_128Concat,
    T::AccountId,
    BoundedVec<UserFileSliceInfo, T::UserFileLimit>,
    ValueQuery,
>;

#[pallet::storage]
#[pallet::getter(fn miner_lock)]
pub(super) type MinerLock<T: Config> =
    StorageMap<_, Blake2_128Concat, AccountOf<T>, BlockNumberFor<T>>;

// ...
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::hooks]
impl<T: Config> Hooks<BlockNumberFor<T>> for Pallet<T> {
    fn on_initialize(now: BlockNumberFor<T>) -> Weight {
        let days = T::OneDay::get();
        let mut weight: Weight = Weight::zero();
        // FOR TESTING
        if now % days == 0u32.saturated_into() {
            let (temp_weight, acc_list) = T::StorageHandle::frozen_task();
            weight = weight.saturating_add(temp_weight);
            let temp_acc_list: BoundedVec<AccountOf<T>, ConstU32<5000>> =
                acc_list.try_into().unwrap_or_default();
            ClearUserList::<T>::put(temp_acc_list);
            weight = weight.saturating_add(T::DbWeight::get().writes(1));
        }

        let mut count: u32 = 0;
        let acc_list = ClearUserList::<T>::get();
        weight = weight.saturating_add(T::DbWeight::get().reads(1));
        for acc in acc_list.iter() {
            // todo! Delete in blocks, and delete a part of each block
            if let Ok(mut file_info_list) = <UserHoldFileList<T>>::try_get(&acc) {
                weight = weight.saturating_add(T::DbWeight::get().reads(1));
                while let Some(file_info) = file_info_list.pop() {
                    count = count.checked_add(1).unwrap_or(ONCE_MAX_CLEAR_FILE);
                    if count == ONCE_MAX_CLEAR_FILE {
                        <UserHoldFileList<T>>::insert(&acc, file_info_list);
                        return weight;
                    }
                    if let Ok(file) = <File<T>>::try_get(&file_info.file_hash) {
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

Pallet Structure: Pallet file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

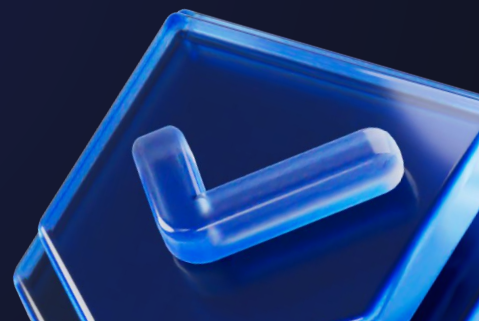
```rust
#[pallet::call]
impl<T: Config> Pallet<T> {
    /// Upload Declaration of Data Storage
    ///
    /// This function allows a user to upload a declaration for data storage, specifying the file's metadata,
    /// deal information, and ownership details. It is used to initiate the storage process of a file.
    ///
    /// Parameters:
    /// - `origin`: The origin of the transaction.
    /// - `file_hash`: The unique hash identifier of the file.
    /// - `deal_info`: A list of segment details for data storage.
    /// - `user_brief`: A brief description of the user and the file's ownership.
    /// - `file_size`: The size of the file in bytes.
    #[pallet::call_index(0)]
    #[transactional]
    #[pallet::weight(<T as pallet::Config>::WeightInfo::upload_declaration(deal_info.len() as u32))]
    pub fn upload_declaration(
        origin: OriginFor<T>,
        file_hash: Hash,
        deal_info: BoundedVec<SegmentList<T>, T::SegmentCount>,
        user_brief: UserBrief<T>,
        file_size: u128,
    ) -> DispatchResult {
        let sender = ensure_signed(origin)?;
        // Check if you have operation permissions.
        ensure!(Self::check_permission(sender.clone(), user_brief.user.clone()), Error::<T>::NoPermission);
        // Check file specifications.
        ensure!(Self::check_file_spec(&deal_info), Error::<T>::SpecError);
        // Check whether the user-defined name meets the rules.
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics

CESS Pallet: file-bank

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

This pallet consists of logics on managing storage space. It allows callers (users) to purchase storage space, expand the purchased space, renew the storage leases. This pallet also implements functions to CRUD (create, read, update, delete) user buckets, a concept similar to user directory. The actual files are segmented and stored in the underlying storage network, but its metadata are stored on-chain.



CESS Pallet: sminer

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

sminer stands for Storage Miner. This pallet contains operations related to storage nodes, allowing them to claim how much space it provides for how long, staking its tokens for its claimed services, and withdrawing the service provision altogether.

CESS

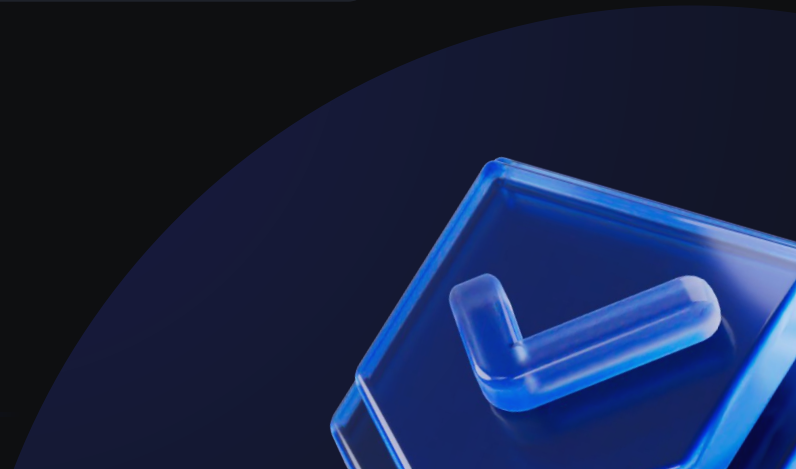
Demo

Adding Your Pallet in CESS Node

Demo: Adding Your Pallet in CESS Node

1. Fork cess Github repo
2. Add a new (template) pallet in the **pallets** directory
3. Add the pallet in the runtime
4. Recompile cess node and execute

Example: cess-examples/cess-node



CESS

Thank you for watching

Please Join Our Community

CESS

CESS Network - Episode 10

Building Custom Pallets

https://www.cess.network