



# CESS Network

The Decentralized Data Infrastructure

## Episode 7: CESS DeOSS and DeOSS REST API



<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



# Ways of Development on CESS



## DIFFICULTY

- |   |     |
|---|-----|
| I. Interacting with CESS using DeOSS API and SDKs   | ★   |
| II. Building and Deploying Smart Contracts          | ★★  |
| III. Building Custom Pallets to Integrate with CESS | ★★★ |



# Why Develop on CESS? CESS Comparison



	CESS	Filecoin	Arweave	Amazon S3
Object Storage Service	✓	✗	✗	✓
CDN	✓	✗	✗	Centralized
OP-Functions	CRUD	CRD	CR	CRUD
Proof Algorithm	Proof of Data Reduplication and Recovery (PoDR <sup>2</sup> ) with Random Challenge	Proof-of-Replication and Proof-of-Space-time with Challenge	Succinct Proof of Random Access	✗
Disaster Recovery	Very High (Guaranteed by PoDR <sup>2</sup> )	Median (A single miner is responsible for the replica integrity)	Median (Storage on the chain)	CRR, SRR, Multi-Destination Replication etc. (incurs additional cost)
Storage Cost	Very Low (\$2-4/T/M)	Low	High (\$4-10/T/M)	Very High (\$21-24/T/M)
Payment Rules	Storage pay, retrieval free	Storage pay, retrieval pay	Storage pay, retrieval free	Storage pay, retrieval pay
Current Capacity	~ 31.1 PiB (Test-net)	~5.98 EiB (current)	~291 PiB (current)	✗
Storage Nodes Number	10834 (June 20th)	2543 (June 20th)	139 (June 20th)	✗



# What is DeOSS?

Decentralized Object Storage Service (DeOSS)

HTTP API Storage Unstructured Data and Metadata Object Retrieval.

- DeOSS is the Gateway to the CESS Network
- DeOSS is the blockchain-based distributed storage protocols
- DeOSS is a Web3 platform for developers
- DeOSS is the decentralized AWS S3



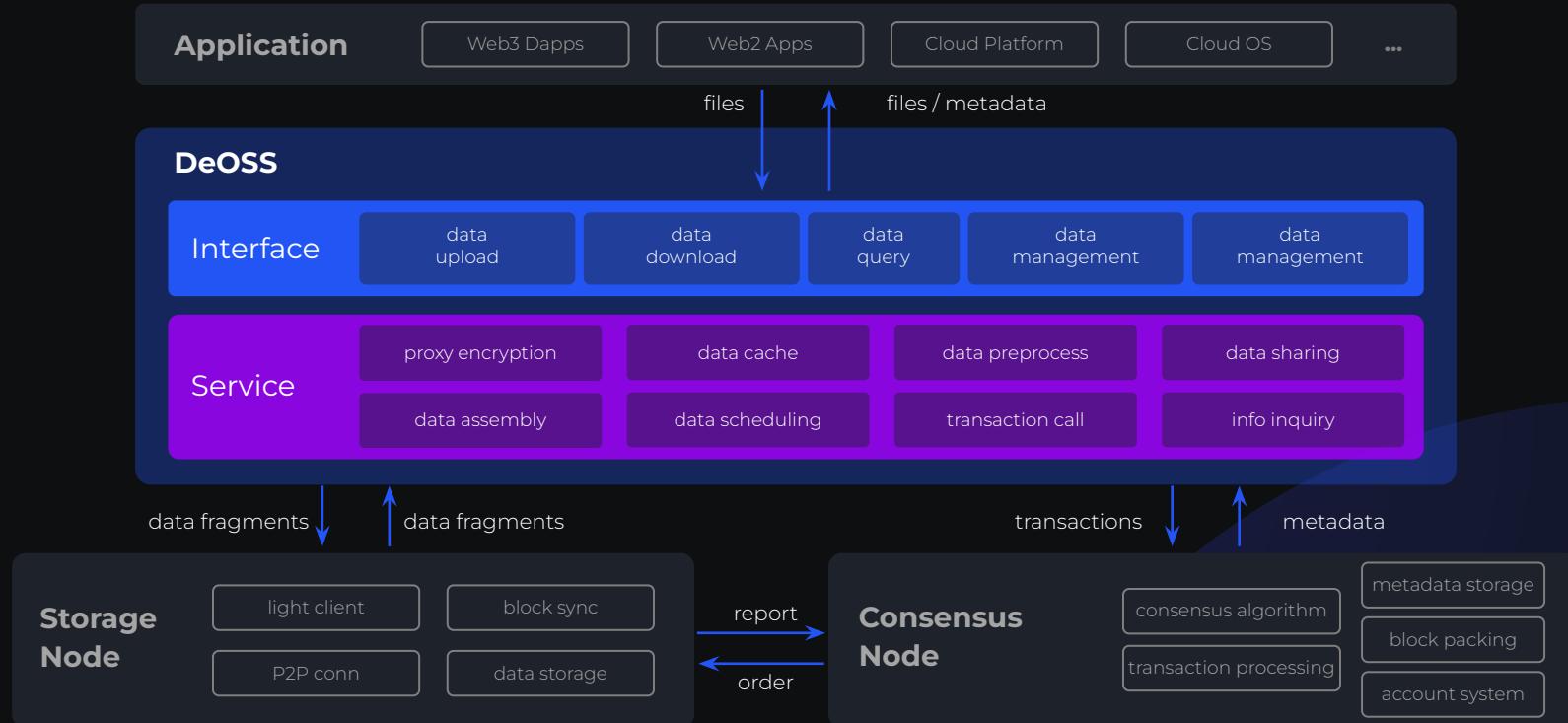
# Why DeOSS?



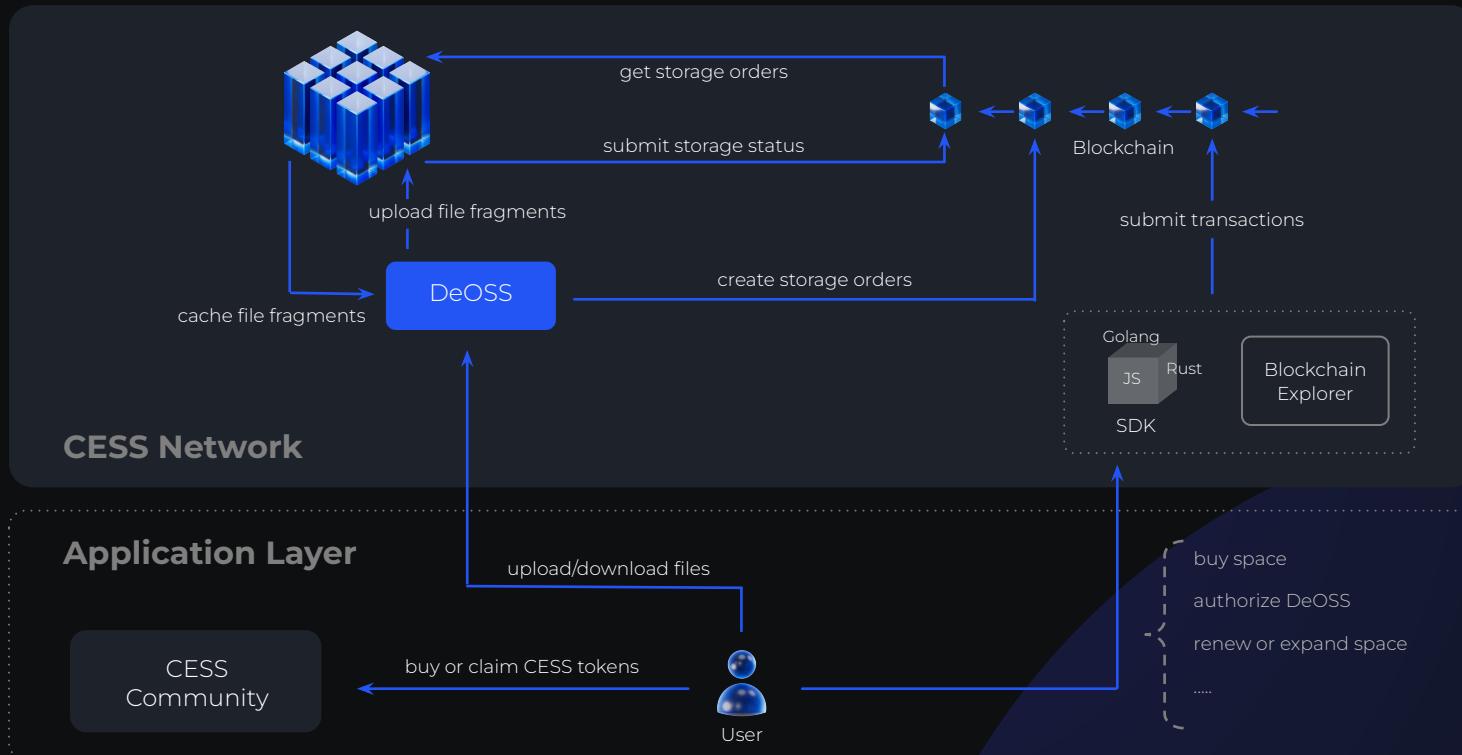
- Effortless Integration with HTTP REST API and SDKs
- Location Based Distributed Data Storage Service
- Highly Scalable
- Lightning-fast Data Access
- Robust Privacy & Data Authorization
- Diverse Data Storage NFT, DA, Backups, etc.
- Data Ownership & Sovereignty
- IPFS Compatible



# DeOSS System Architecture

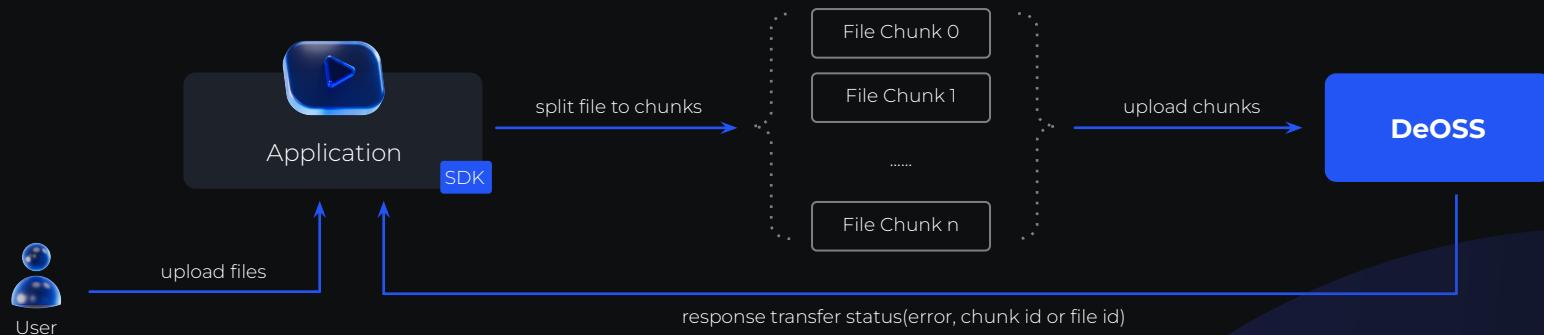


# Data Access Process: DeOSS Workflow



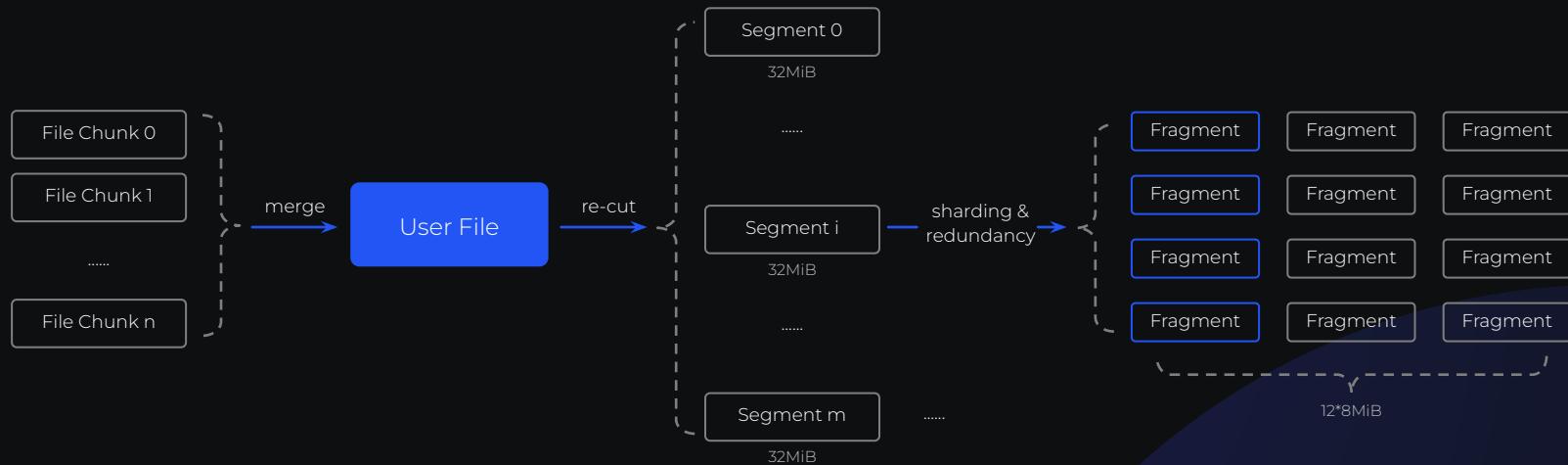
# Data Access Process: File Upload

## 1. Resumable Chunk Upload



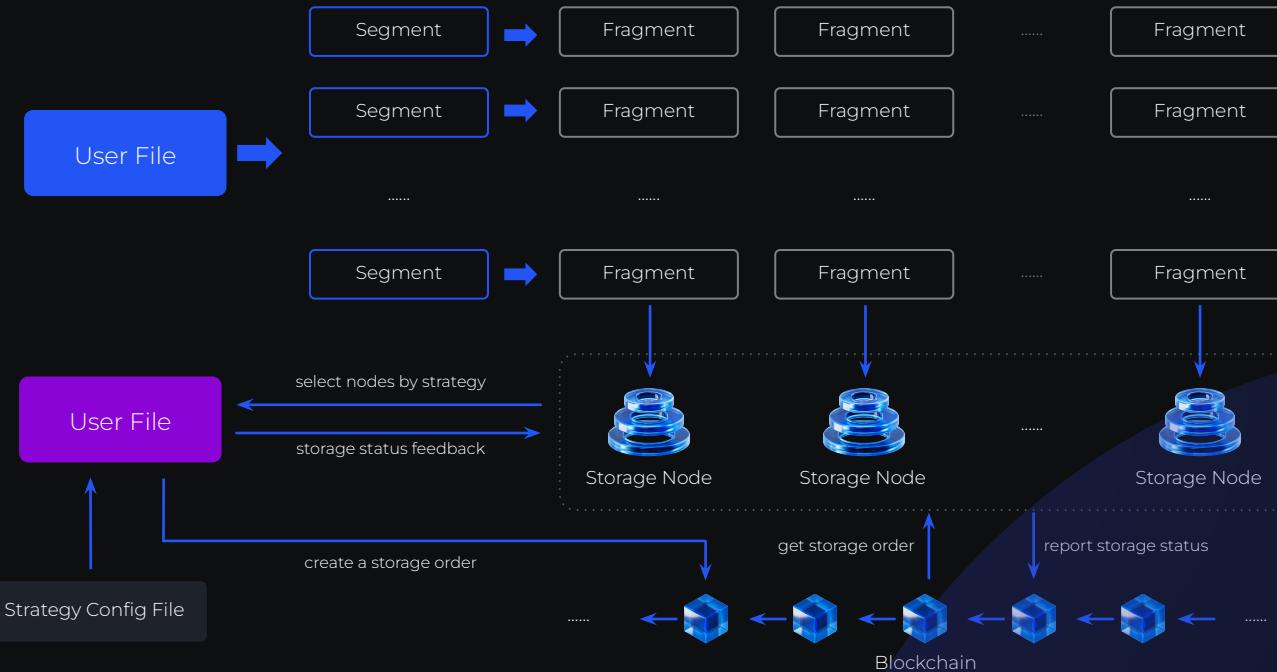
# Data Access Process: File Upload

## 2. Data Processing



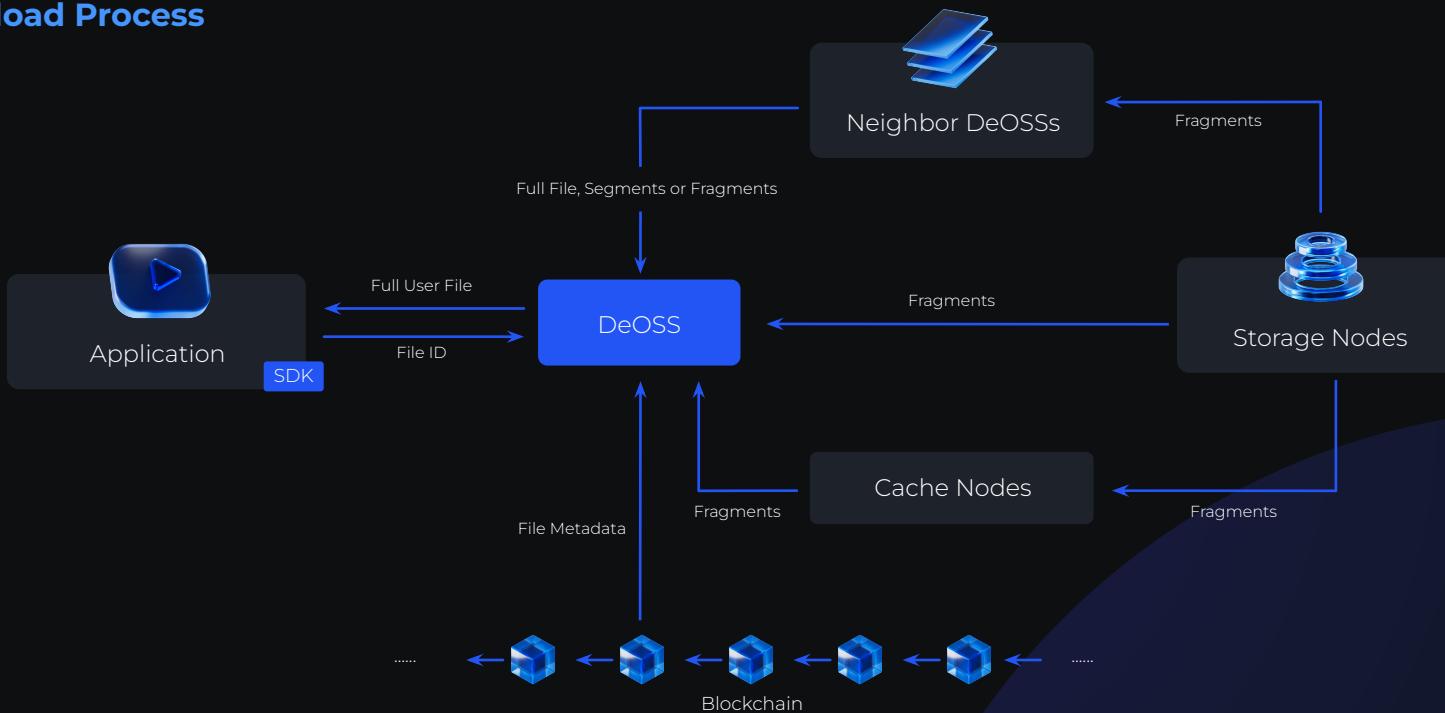
# Data Access Process: File Upload

## 3. Data Distribution



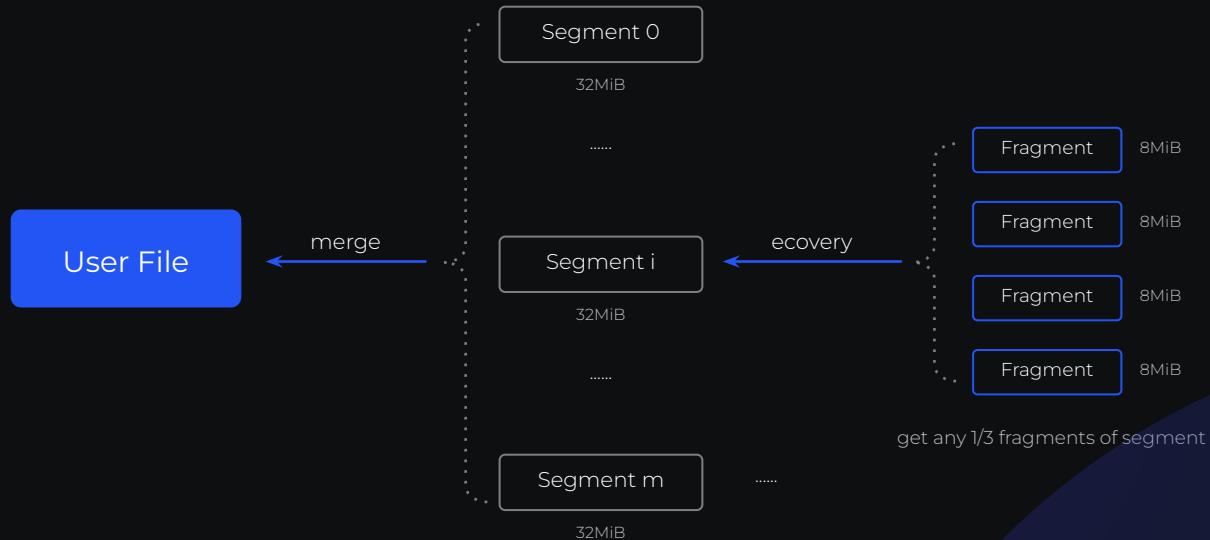
# Data Access Process: File Download

## 1. Download Process



# Data Access Process: File Download

## 2. File Merging



# DeOSS File Encryption and Decryption



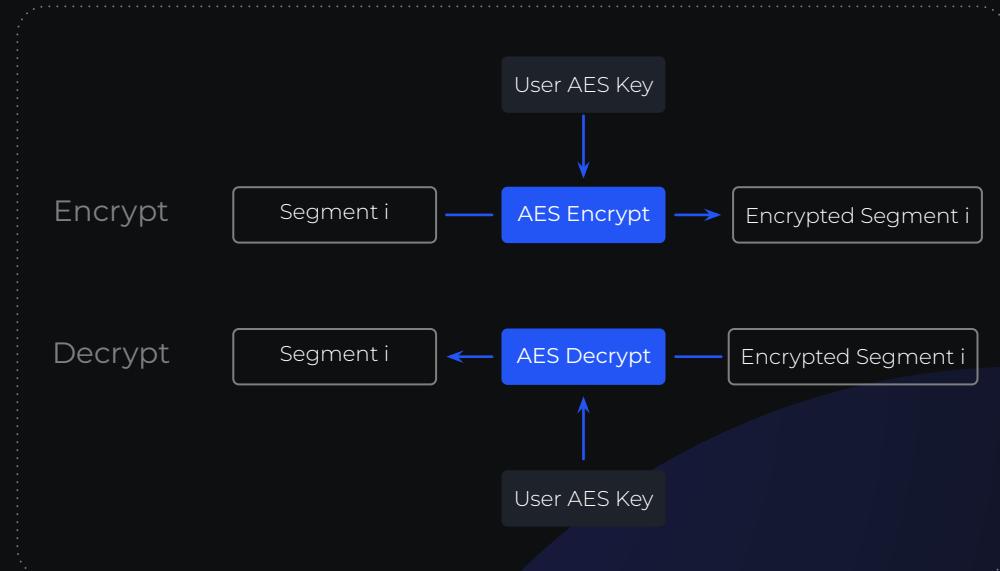
- AES Symmetric Key Encryption
- Proxy Re-Encryption

## Encryption During Pre-processing

File is segmented and each segment is encrypted individually

## Decryption During File Merging

Each Segment is Decrypted and Merged



Rust

GoLang

Javascript

RESTFul API



# DeOSS Demo

## Using REST API





# DeOSS REST API Demo

## Prerequisites

### CESS Wallet Account

AND

### Public DeOSS Gateways

<https://deoss-pub-gateway.cess.network/> cXhwBytXqrZLr1qM5NHJhCzEMckSTzNKw17ci2aHft6ETSQm9

<https://deoss-sv.cess.network> cXjy16zpi3kFU6ThDHeTifpwHop4YjaF3EvYipTeJSbTjmayP

<https://deoss-fra.cess.network> cXhkf7fFTToo8476oeRqxyWVnxF8ESsd8b7Yh258v6n26RTkL

<https://deoss-sgp.cess.network> cXf3X3ugTnivQA9iDRYmLNzxSqybgDtpStBjFcBZEoH33UVaz

OR

### Self Hosted DeOSS Gateway

[Setting up your own DeOSS Gateway](#)

# DeOSS REST API Demo



## 1. Purchase Space

Navigate to [CESS Explorer](#), Developer > Extrinsics

The screenshot shows the CESS Explorer interface with the following steps highlighted:

1. Select Extrinsic: A dropdown menu is open, with "Developer" selected.
2. Select wallet account: The account "USER1 (EXTENSION)" is selected.
3. Select storageHandler: The option "storageHandler" is selected from a dropdown.
4. Select mintTerritory: The territory "mintTerritory(gibCount, territoryName)" is selected.
5. Fill in the size of the territory in Gib: The value "10" is entered in the "gibCount: u32" field.
6. Fill in the territory name: The territory name "myTerritory" is entered in the "territoryName: Bytes (TerrName)" field.
7. Submit transaction: Two buttons are shown: "Submit Unsigned" and "Submit Transaction". The "Submit Transaction" button is highlighted with a red box.

Other visible details include:

- Header: "cess-devnet", "cess-node/100 #113,959", "Accounts", "Network", "Governance", "Developer", "Settings", "GitHub", "Wiki", "CESS Node v0.7.7 api v1.0.2 apps v0.141.3".
- Left sidebar: "Extrinsics", "Submission", "Decode".
- Bottom left: "encoded call data" and "encoded call hash".
- Bottom right: "encoding details" showing "callindex 6b00", "gibcount 0x000000", "territoryname 2c 6d795465727269746f7279", and a "link" to "#/extrinsics/decode/0x6b000a0000002c6d795465727269746f7279".

- Select the Desired Account in “using the selected account”
- Select “storageHandler” and “buySpace(gibCount)” from “submit the following extrinsic”
- Enter the amount of space you would like to purchase in “gibCount: u32”
- Click on “Submit Transaction”, and Sign and Submit the transaction.

# DeOSS REST API Demo



## 2. Authorize Gateway

Navigate to [CESS Explorer](#), Developer > Extrinsics

The screenshot shows the CESS Explorer interface with the following steps highlighted:

- 1. Select Extrinsic**: The first step is to select the extrinsic type, which is highlighted with a red box.
- 2. Select wallet account**: The selected account "USER1 (EXTENSION)" is highlighted with a red box.
- 3. Select oss**: The "oss" option is highlighted with a red box.
- 4. Select authorize**: The "authorize(operator)" option is highlighted with a red box.
- 5. Fill in the gateway wallet account you are using**: The "operator: AccountId32 (AccountOf)" field contains "USER1 (EXTENSION)" and is highlighted with a red box.
- 6. Submit transaction**: The "Submit Transaction" button is highlighted with a red box.

Other visible details include:

- Accounts: Accounts dropdown.
- Network: Network dropdown.
- Governance: Governance dropdown.
- Developer: Developer dropdown (highlighted).
- Settings: Settings icon.
- Github: GitHub link.
- Wiki: Wiki link.
- CESS Node: v0.77, api: v12.0.2, apps: v0.141.1.

- Select the Desired Account in “using the selected account”
- Select “oss” and “authorize(operator)” from “submit the following extrinsic”
- Enter the wallet address of Public DeOSS Gateway or Self Hosted Gateway in “operator: AccountId32(AccountOf)”
- Click on “Submit Transaction”, and Sign and Submit the transaction.

# DeOSS REST API Demo



## 3. Generate Identity Signature

Navigate to [CESS Explorer](#), Developer > Sign and Verify

The screenshot shows the CESS Explorer interface with the 'Sign and Verify' tab selected. Key elements highlighted with red boxes include:

- The 'Developer' menu item in the top navigation bar.
- The 'Sign and verify' tab in the main content area.
- The 'account' dropdown showing 'CESS TEST (EXTENSION)'.
- The 'sign the following data' input field containing 'hello'.
- The 'Sign message' button at the bottom right.

The 'signature of supplied data' field displays the hex string: 0x28846be05957197c3e3cbf686fcfcbaef2c6e6d13f8405a1fe420d235ee0638304ef0f8f2c89f2894d3128779d2ef00a06e4088052d43585c20969f329fa22f82.

- Select the Desired Account in “account” Section
- Enter a message in “sign the following data”
- Click on “Sign Message”, enter your account password and click on “Sign the message”
- Copy the signature from “signature of supplied data”. This will be used in later section.

# DeOSS REST API Demo



## 4. Uploading a File

From your Terminal execute

```
curl -X PUT https://deoss-pub-gateway.cess.cloud/ -F 'file=<FILE_PATH>;type=<FILE_TYPE>' -H "Account: <CESS_ACCOUNT_ADDRESS>" -H "Message: <YOUR_MESSAGE>" -H "Signature: <YOUR_SIGNATURE>" -H "BucketName: <BUCKET_NAME>" -H "Territory: <TERRITORY_NAME>"
```

- You can replace Public DeOSS Gateway URL with your DeOSS URL if you have setup DeOSS Gateway yourself
- Replace <FILE\_PATH> "/root/text.log" with the absolute file path and <FILE\_TYPE> with the desired file type "application/octet-stream"
- Replace <CESS\_ACCOUNT\_ADDRESS> with the address from which you purchased space
- Enter the same message you entered while generating signature
- Replace <YOUR\_MESSAGE> with the message you entered in the last step
- Replace <YOUR\_SIGNATURE> with the signature you generated in the last step
- Replace <BUCKET\_NAME> with the name of your choice and <TERRITORY\_NAME> with the name of the Territory you used while purchasing space

# DeOSS REST API Demo



## 5. Downloading File

From your Terminal execute

```
curl -X GET -o samplefile https://deoss-pub-gateway.cess.cloud/<FID> -H "Operation: download" -H "Account: <CESS_ACCOUNT_ADDRESS>"
```

- Replace FID with the FileID you received in the previous step
- Replace CESS\_ACCOUNT\_ADDRESS with the address from which you purchased space

# DeOSS REST API Demo



## 6. Deleting a File

From your Terminal execute

```
curl -X DELETE https://deoss-pub-gateway.cess.cloud/<FID> -H "Account: <CESS_ACCOUNT_ADDRESS>" -H "Message: hello" -H "Signature: <YOUR_SIGNATURE>"
```

- Replace FID with the FileID you received in the previous step
- Replace CESS\_ACCOUNT\_ADDRESS with the address from which you purchased space
- Replace <YOUR\_MESSAGE> with the message you entered in the previous step
- Replace YOUR\_SIGNATURE with the signature you generated in the previous step

# DeOSS REST API Demo



## 7. Other Commands

### Get Object Info

```
curl -X GET <DEOSS_URL>/<FID> -H "Operation: view"
```

### Creating a Bucket

```
curl -X PUT <DEOSS_URL>/ -H "BucketName: <BUCKET_NAME>" -H "Account: <CESS_ACCOUNT>" -H "Message: xxx" -H "Signature: <CESS_ACCOUNT>"
```

### Fetching a Bucket

```
curl -X GET <DEOSS_URL>/<BUCKET_NAME> -H "Account: <CESS_ACCOUNT>"
```

### List Bucket

```
curl -X GET <DEOSS_URL>/* -H "Account: <CESS_ACCOUNT>"
```

### Deleting Bucket

```
curl -X DELETE <DEOSS_URL>/<BUCKET_NAME> -H "Account: <CESS_ACCOUNT>" -H "Message: <CESS_ACCOUNT>" -H "Signature: <SIGNATURE>"
```



# Thank you for watching

Please Join Our Community





# CESS Network - Episode 7

CESS DeOSS and DeOSS REST API



<https://www.cess.network>



# What is DeOSS?

Decentralized Object Storage Service (DeOSS) is a decentralized object-based mass storage service that provides low-cost, secure, and scalable distributed data storage services.

- Ideal for handling diverse data types
- Provides data infrastructure for AI applications
- As efficient as AWS S3, as developer friendly as Google Cloud and as smooth as TikTok
- Designed for low latency and high throughput
- Offers a convenient platform for developing applications
- Authorization Management
- Custom Development Capabilities
- Ensures Data Ownership



# Why and Who?



## Why DeOSS?

- HTTP API for Storing Unstructured Data and Retrieving Metadata and Objects
- Location Based Distributed Data Storage Service
- Object-based Storage
- Privacy & Data Authorization
- Diverse Data Storage NFT, DA, Backups, etc.
- Decentralization
- Highly Scalable
- Data Ownership Protection
- Anti-censorship
- Millisecond Data Indexing

## Who can use DeOSS?

- DApp
- NFT
- Apps Requiring Large-Scale Data Storage
- Web2 & Web3 Developers

