

# Eluvio Content Fabric V2 Spec

Eluvio

2022

## 0 Definitions

**Node** A server which stores and serves parts.

**Provider** An individual or organization which owns, secures, and operates nodes.

**Tenant** An individual or organization which owns content.

**Content** A piece of data which is owned by a tenant.

**Space** A group of providers, provider nodes, and tenants, where providers agree to run nodes which store content owned by a tenant according to a common set of rules.

**Part** A part is a sequence of bytes stored in the space, referenced by its hash.

**Content Object Version** A collection of parts created by a tenant, referenced by its hash.

**Content Object** A collection of versions.

**Library** A 'folder' of content objects owned by a tenant with a permission structure that determines who within a tenancy is able to create/modify/delete content objects and content object versions.

**KMS** A tenant-owned server which holds keys for encrypting/decrypting content which the tenant stores in the space.

**Blockchain** A distributed ledger responsible for orchestrating cooperation between providers, the exchange of value between providers and tenants, and governance that determines the rules of a space.

The following entities are defined by fixed length identifiers as follows:

Entity	Identifier	Type
Node	ID <sub>node</sub>	Fixed length byte string
Space	ID <sub>space</sub>	Fixed length byte string
Content Object	ID <sub>conq</sub>	Fixed length byte string
Content Object Version	ID <sub>version</sub>	Fixed length byte string
KMS	ID <sub>kms</sub>	Fixed length byte string
Library	ID <sub>lib</sub>	Fixed length byte string

## 1 Spaces

The space functions as the top level governance structure of the fabric that orchestrates how providers cooperate to serve tenant data. Governance is TBD.

### 1.1 Space rules

**Provider Bond** An amount,  $BOND_{prov}$ , of currency each provider must lock up in order to participate within the space. Funds can be slashed from here if a provider misbehaves.

**Tenant Bond** An amount,  $BOND_{ten}$ , of currency each tenant must lock up in order to participate within the space. Funds can be slashed from here if a tenant misbehaves.

**SLAs** Specifications for availability requirements provider nodes must have.

**Partition number** The partitioning constant for part storage

## 2 Providers

A provider is a logical group of nodes within a space, and a permission structure for keys.

### 2.1 Provider Permissions

Provider keys have the following permission levels, from most to least privileged

1.  $PERM_{root}$  Root level
  - add/remove admins (effectively allows for admin key rotation)
2.  $PERM_{admin}$  Admin level
  - add/remove nodes
  - bill tenants
3.  $PERM_{node}$  Node level
  - Co-author versions with tenants

- Mark itself as no longer pending
- Participate in part networking

## 2.2 Blockchain actions

In addition to setting permissions on keys, we have the following actions

**CreateProvider**( $k_{\text{origin}} = k_{\text{root}}, \text{ID}_{\text{space}}, \text{ID}_{\text{prov}}$ ) Creates the provider

- TODO: check governance to see whether origin can create a provider
- Creates  $\text{ID}_{\text{prov}}$  and sets its space to  $\text{ID}_{\text{space}}$
- Sets  $k_{\text{root}}$  as the creator of  $\text{ID}_{\text{prov}}$
- Sets  $k_{\text{root}}$  as a key for  $\text{ID}_{\text{prov}}$  with level  $\text{PERM}_{\text{root}}$
- Bonds  $\text{BOND}_{\text{prov}}$  from  $k_{\text{root}}$  to the space under  $\text{ID}_{\text{prov}}$

**AddNode**( $k_{\text{origin}}, \text{ID}_{\text{prov}}, \text{ID}_{\text{node}}, k_{\text{node}}, \text{LOC}_{\text{node}}$ ) adds a node

- Checks that  $k_{\text{origin}}$  has permission  $\text{PERM}_{\text{admin}}$  or above for  $\text{ID}_{\text{prov}}$ .
- Creates a node  $\text{ID}_{\text{node}}$  with locator  $\text{LOC}_{\text{node}}$
- Registers  $k_{\text{node}}$  to  $\text{ID}_{\text{prov}}$  with permission level  $\text{PERM}_{\text{node}}$ <sup>1</sup>
- Marks the node as pending while it syncs up parts with the rest of the space

**ConfirmNode**( $k_{\text{origin}}, \text{ID}_{\text{prov}}, \text{ID}_{\text{node}}$ ) marks a node as no longer pending

- Checks that  $k_{\text{origin}}$  has permissions  $\text{PERM}_{\text{node}}$  or above for  $\text{ID}_{\text{prov}}$
- Sets  $\text{ID}_{\text{node}}$  to no longer pending

**RemoveNode**( $k_{\text{origin}}, \text{ID}_{\text{prov}}, \text{ID}_{\text{node}}$ ) removes a node

- Checks that  $k_{\text{origin}}$  has permissions  $\text{PERM}_{\text{admin}}$  or above for  $\text{ID}_{\text{prov}}$
- Removes all  $\text{ID}_{\text{node}}$  information from the space and provider

**BillTenant** TODO

## 3 Tenants

A tenant is an owner of content, responsible for providing a service which manages keys which encrypt content to providers.

---

<sup>1</sup>Should this error if the key already exists within the permissions scheme?

### 3.1 Tenant Permissions

Tenant keys have the following permission levels, from most to least privileged

1.  $\text{PERM}_{\text{root}}$  can add/remove admins
2.  $\text{PERM}_{\text{admin}}$  can add/remove kmses, add/remove users to/from libraries
3.  $\text{PERM}_{\text{kms}}$  can co-author content object versions with provider nodes

### 3.2 Blockchain Actions

**CreateTenant**( $k_{\text{origin}} = k_{\text{root}}, \text{ID}_{\text{space}}, \text{ID}_{\text{tenant}}$ ) creates a tenancy

- TODO: check governance to see whether origin can create a tenant
- Creates  $\text{ID}_{\text{tenant}}$  and sets its space to  $\text{ID}_{\text{space}}$
- Sets  $k_{\text{root}}$  as the creator of  $\text{ID}_{\text{tenant}}$
- Sets  $k_{\text{root}}$  as a key for  $\text{ID}_{\text{tenant}}$  with level  $\text{PERM}_{\text{root}}$
- Bonds  $\text{BOND}_{\text{ten}}$  from  $k_{\text{root}}$  to the space under  $\text{ID}_{\text{tenant}}$

**AddKMS**( $k_{\text{origin}}, \text{ID}_{\text{tenant}}, \text{ID}_{\text{kms}}, k_{\text{kms}}, \text{LOC}_{\text{kms}}$ ) creates a kms

- Checks that  $k_{\text{origin}}$  has permission  $\text{PERM}_{\text{admin}}$  or above for  $\text{ID}_{\text{tenant}}$ .
- Creates a node  $\text{ID}_{\text{kms}}$  with locator  $\text{LOC}_{\text{kms}}$
- Registers  $k_{\text{kms}}$  to  $\text{ID}_{\text{tenant}}$  with permission level  $\text{PERM}_{\text{kms}}$

**RemoveKMS**( $k_{\text{origin}}, \text{ID}_{\text{tenant}}, \text{ID}_{\text{kms}}$ ) removes a node

- Checks that  $k_{\text{origin}}$  has permissions  $\text{PERM}_{\text{admin}}$  or above for  $\text{ID}_{\text{tenant}}$
- Removes all  $\text{ID}_{\text{kms}}$  information from the space and tenancy

**TODO: Remove Tenant, Top up billing balance**

## 4 Key Management Services (KMSs)

## 5 Content Objects

## 6 Part networking

TODO: @Serban