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	$\texttt{ID}_{\texttt{node}}$	Fixed length byte string
Space	$\texttt{ID}_{\texttt{space}}$	Fixed length byte string
Content Object	$\texttt{ID}_{\texttt{conq}}$	Fixed length byte string
Content Object Version	$\texttt{ID}_{\texttt{version}}$	Fixed length byte string
KMS	$\texttt{ID}_{\texttt{kms}}$	Fixed length byte string
Library	$\texttt{ID}_{\texttt{lib}}$	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, $\texttt{BOND}_{\texttt{prov}}$, 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, $\texttt{BOND}_{\texttt{ten}}$, 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. $\texttt{PERM}_{\texttt{root}}$ Root level
   - add/remove admins (effectively allows for admin key rotation)
2. $\texttt{PERM}_{\texttt{admin}}$ Admin level
   - add/remove nodes
   - bill tenants
3. $\texttt{PERM}_{\texttt{node}}$ Node level
   - Co-author versions with tenants

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

2.2 Provider blockchain actions

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

CreateProvider($k_{\texttt{origin}}, \texttt{ID}_{\texttt{space}}, \texttt{ID}_{\texttt{prov}}$) Creates the provider

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

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

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

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

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

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

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

BillTenant TODO

3 Tenants

A tenant is an owner and creator of content. They are responsible for providing a service available to providers' nodes which can manage keys and encrypt/decrypt content.

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

3.2 Tenant Blockchain Actions

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

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

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

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

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

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

TODO: Remove Tenant, Top up billing balance

4 Libraries

Libraries group keys together which may create/modify content within a specific context. They act as a way to separate the keys which manage tenant infrastructure (like KMSs) from keys which manage content.

4.1 Library Rights

Any `user` in a library has some associated rights, $\texttt{rights}_{\texttt{user}} \in \mathcal{R}$. The exact format of this structure is TDB (currently they're bitflags), but should at the very least be able to answer the following questions:

$\texttt{IsAdmin}(\texttt{rights}_{\texttt{user}})$**:** Is the user allowed to add other users as non-admins

$\texttt{CanEdit}(\texttt{rights}_{\texttt{user}})$**:** Is the user allowed to edit content

4.2 Library Blockchain Actions

CreateLibrary($k_{\texttt{origin}}, \texttt{ID}_{\texttt{tenant}}, \texttt{ID}_{\texttt{lib}}, \text{name}$) creates a library

- Checks the origin has permission $\texttt{PERM}_{\texttt{admin}}$ within $\texttt{ID}_{\texttt{tenant}}$
- Creates a library with $\texttt{ID}_{\texttt{lib}}$ and the given name within $\texttt{ID}_{\texttt{tenant}}$
- Initialize the tenant as a library admin by setting $\texttt{rights}_{\texttt{origin}}$ such that $\texttt{IsAdmin}(\texttt{rights}_{\texttt{origin}})$ is true

SetRights($k_{\texttt{origin}}, \texttt{ID}_{\texttt{tenant}}, \texttt{ID}_{\texttt{lib}}, k_{\texttt{target}}, \texttt{rights}_{\texttt{target}}$)

5 Content Objects

6 Key Management Services (KMSs)

7 Part networking

TODO: @Serban