Large Object API

Namazu Elements Core Features · https://namazustudios.com/docs/namazu-elements-core/large-object-api/

A **Large Object** is arbitrary file content (an image, a data blob, a downloadable asset) stored and managed by Elements as a first-class database record, with its own id, permissions, and lifecycle. It's the mechanism behind the **Large Object** section of the admin console (see CMS Feature Overview) and is also fully available over the REST API, which makes it a convenient place to store game assets you want to manage the same way you manage other Elements data: live ops content, A/B test variants, user-uploaded images, and similar.

Note

This is a different system from Element Static Content and Dashboard UI Plugins. Large Object content is stored as database content addressed by object id, not as a file tree tied to an Element or an Application.

The Large Object Model

Field	Description
`id`	Unique identifier for the object.
`path`	A logical path used for search/organization. If you don't supply one when creating an object, an automatic path is generated from the object's MIME type and a random id.
`mimeType`	The MIME type of the stored content.
`state`	Either `EMPTY` (created but no content uploaded yet) or `UPLOADED` (content has been written).
`accessPermissions`	Read, write, and delete permissions for the object. See below.
`lastModified`	Timestamp of the last metadata or content change.
`originalFilename`	The filename supplied at upload time, if any.

Field	Description
`url`	Computed, publicly reachable URL for the object's content. Not stored; assembled on read. See Serving Content below.

Object content itself (the actual bytes) is stored separately from the metadata above, in MongoDB GridFS, keyed by the object's id.

Access Permissions

Each Large Object carries its own `accessPermissions`, with a separate permission for reading, writing, and deleting the object. Each of these can either be a wildcard, meaning "anyone," or scoped to specific subjects. A publicly downloadable game asset would have a wildcard read permission with write and delete restricted; a private user upload would have all three restricted.

Access Levels

Access level	Capabilities
Anonymous	Can list objects and read metadata. Can read an object's content only if its read permission is a wildcard. Create, update, delete, and non-public reads are all forbidden.
User	Can read, update, and delete objects it has been granted permission for, per that object's `accessPermissions`. Cannot create new Large Objects; creation is Superuser-only.
Superuser	Full create, read, update, and delete access to any object, plus the ability to create an object by fetching content from a URL server-side rather than uploading it directly.

REST API

Method	Path	Description
`POST`	`/large_object`	Creates a new Large Object record (metadata only; content is uploaded separately). Superuser only.

Method	Path	Description
`POST`	`/large_object/from_url`	Creates a new Large Object by having Elements fetch content from a supplied URL server-side. Superuser only.
`PUT`	`/large_object/{id}`	Updates an object's metadata and/or access permissions.
`PUT`	`/large_object/{id}/content`	Uploads (or replaces) an object's content as a multipart request body.
`GET`	`/large_object/{id}`	Gets a single object's metadata.
`GET`	`/large_object?offset=&count=&search=`	Lists objects, paginated, optionally filtered by a search term matched against `path` and `mimeType`.
`DELETE`	`/large_object/{id}`	Deletes an object's metadata and content.

Serving Content

A Large Object's public `url` field is assembled as the configured CDN origin plus `/object/{id}`:

```
{dev_getelements_elements_cdn_url}/object/{id}
```

`dev_getelements_elements_cdn_url` is the same CDN origin variable described in Configuring External URLs for Deployment. Requests to this endpoint are resolved through the same access-level rules described above (so a private object still returns `403` to an unauthorized caller), and support:

- Conditional requests via `ETag` / `If-None-Match`, computed from the object's id and last-modified time.
- `Cache-Control: public, max-age=...` (using the `dev.getelements.elements.cdn.public.max.age` attribute, default 300 seconds) when the object's read permission is a wildcard, and `private, no-store` otherwise.
- A `?download=true` query parameter, which sends `Content-Disposition: attachment` instead of the default inline disposition.

Events

Large Object creation, updates, and deletion publish events that an Element can subscribe to, the same way most other data types in Elements do. See the 3.9 release notes for background on this event coverage.