AzureBlobService

September 4, 2026

AzureBlobService

Kind: Service

Source: `atloria-monorepo/apps/api/src/assets/azure-blob.service.ts`

`AzureBlobService` is a NestJS backend service responsible for managing file assets stored in Azure Blob Storage. It uploads blobs, deletes existing assets, builds public blob URLs, and generates time-limited SAS URLs for controlled client access.

Methods

Method	Signature	Returns	Description
`upload`	`upload(options: {		

```
buffer: Buffer;
filename: string;
mimeType: string;
organizationId: string;
/** Optional sub-folder under the org (e.g. 'videos' for D1 video artifacts). */
prefix?: string;
```

}) | Promise<{ url: string; cdnUrl: string }> | Upload image to Azure Blob Storage | | delete | delete(blobName: string) | Promise | Delete blob (soft delete recommended) | | getBlobUrl | getBlobUrl(blobName: string) | string | Get blob URL | | generateSasUrl | generateSasUrl(blobName: string, _expiresInMinutes: unknown) | Promise` | Generate SAS token for temporary access (if needed for private assets) TODO: Implement SAS token generation with proper credentials |

Dependencies

- `ConfigService`

When something fails

- `AzureBlobService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Client
  participant API as NestJS Controller
  participant Service as AzureBlobService
  participant Azure as Azure Blob Storage
  participant CDN as CDN

  Client->>API: Upload asset
  API->>Service: upload(file, options)
  Service->>Azure: Upload blob
  Azure-->>Service: Blob stored
  Service->>Service: getBlobUrl(blobName)
  Service-->>API: { url, cdnUrl }
  API-->>Client: Asset URLs

  Client->>API: Request temporary download URL
  API->>Service: generateSasUrl(blobName)
  Service->>Azure: Create SAS token
  Azure-->>Service: Signed access URL
  Service-->>API: SAS URL
  API-->>Client: Temporary URL
```

Usage

```
import { Controller, Delete, Param, Post, UploadedFile, UseInterceptors } from '@nestjs/common'
import { FileInterceptor } from '@nestjs/platform-express';
import { AzureBlobService } from './azure-blob.service';

@Controller('assets')
export class AssetsController {
  constructor(private readonly azureBlobService: AzureBlobService) {}

  @Post()
  @UseInterceptors(FileInterceptor('file'))
  async uploadAsset(@UploadedFile() file: Express.Multer.File) {
    const { url, cdnUrl } = await this.azureBlobService.upload(file);

    return {
      originalUrl: url,
      deliveryUrl: cdnUrl,
    };
  }
```

```
  @Delete(':blobName')
  async deleteAsset(@Param('blobName') blobName: string): Promise<void> {
    await this.azureBlobService.delete(blobName);
  }

  @Post(':blobName/sas-url')
  async createTemporaryAccessUrl(@Param('blobName') blobName: string) {
    return {
      url: await this.azureBlobService.generateSasUrl(blobName),
    };
  }
}
```

AI Coding Instructions

- Inject and use `AzureBlobService` from controllers or domain services; keep Azure Storage SDK calls centralized in this service.
- Store the returned `cdnUrl` for client-facing asset delivery when a CDN is configured, while retaining the blob URL when direct storage access is required.
- Use `generateSasUrl()` for private blobs instead of exposing storage credentials or constructing SAS tokens in controllers.
- Validate uploaded file types, sizes, and generated blob names before calling `upload()`; do not trust client-provided filenames directly.
- Ensure delete operations use the exact stored blob name and handle missing blobs or Azure storage failures appropriately.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `AssetsModule` (MODULE_PROVIDES)
- `AssetsModule` (MODULE_EXPORTS)
- `AssetsService` (DEPENDS_ON)
- `DocVersionExportService` (DEPENDS_ON)