Files Management

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2025-01-05
Version: v1.0

Base URL: `https://api.fursahub.com/api/v1`

Short Description: File upload and management endpoints for Fursa Hub. Handles image uploads for profiles, posts, events, and other content. Uses MinIO for storage with automatic BlurHash generation for images.

Hints:

- Maximum file size: 25MB per file
- Supported image types: JPEG, PNG, GIF, WebP, BMP
- Supported video types: MP4, AVI, MOV, WebM, MKV
- Supported documents: PDF, Word, Excel, Text files
- Each user gets their own storage bucket
- BlurHash automatically generated for images (use for loading placeholders)
- Files are publicly accessible via permanentUrl

File Storage Architecture

```
┌──────────────────────────────────────────────────────────────────────────┐
│                        FILE STORAGE ARCHITECTURE                         │
├──────────────────────────────────────────────────────────────────────────┤
│                                                                          │
│  UPLOAD FLOW:                                                            │
│  ┌──────────────────────────────────────────────────────────────────┐    │
│  │ 1. Client uploads file to: POST /files/upload-single             │    │
│  │ 2. Backend:                                                      │    │
│  │    ├── Validates file (size, type)                               │    │
│  │    ├── Creates user bucket if needed (fursa-{userId})            │    │
│  │    ├── Generates unique filename (UUID)                          │    │
│  │    ├── Uploads to MinIO storage                                  │    │
│  │    ├── If image: generates BlurHash + dimensions                 │    │
│  │    └── Returns file metadata with permanentUrl                   │    │
```

```
│  └──────────────────────────────────────────────────────────────┘     │
│                                                                      │
│  STORAGE STRUCTURE:                                                  │
│  MinIO Server                                                        │
│  └── fursa-{userId}/                  ← User's bucket                │
│      ├── profile/                     ← Profile photos               │
│      │   └── abc123-uuid.jpg                                         │
│      ├── social_post/                 ← Post media                   │
│      │   ├── def456-uuid.jpg                                         │
│      │   └── ghi789-uuid.mp4                                         │
│      ├── events/                      ← Event images                 │
│      ├── opportunities/               ← Opportunity attachments      │
│      ├── calls/                       ← Call for proposals           │
│      ├── funds/                       ← Fund documents               │
│      ├── innovation/                  ← Innovation hub files         │
│      └── skill_center/                ← Course materials             │
│                                                                      │
│  ACCESS:                                                             │
│  └── Files accessible at: {filesServerUrl}/{bucketName}/{objectKey}  │
│  └── Example: https://files.fursahub.com/fursa-uuid123/profile/pic.jpg │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

File Directories

Directory	Purpose	Typical Use
`PROFILE`	Profile photos	User avatar, cover images
`SOCIAL_POST`	Social media posts	Post images, videos
`CALLS`	Call for proposals	Proposal documents
`FUNDS`	Funding/grants	Grant application files
`EVENTS`	Events	Event banners, tickets
`OPPORTUNITIES`	Job/opportunity	Job post attachments
`INNOVATION`	Innovation hub	Project files
`SKILL_CENTER`	Skills/courses	Course materials

Endpoints

1. Upload Single File

Purpose: Upload a single file to the specified directory.

Endpoint: `POST` `{base_url}/files/upload-single`

Access Level: Protected

Authentication: Bearer Token

Content-Type: `multipart/form-data`

Request Parameters:

Parameter	Type	Required	Description	Validation
file	file	Yes	File to upload	Max 25MB
directory	string	Yes	Target directory	enum: PROFILE, SOCIAL_POST, CALLS, FUNDS, EVENTS, OPPORTUNITIES, INNOVATION, SKILL_CENTER

Example Request (using curl):

```
curl -X POST \
  -H "Authorization: Bearer {accessToken}" \
  -F "file=@/path/to/image.jpg" \
  -F "directory=PROFILE" \
  {base_url}/files/upload-single
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "File uploaded successfully",
  "action_time": "2025-01-05T10:30:45",
  "data": {
```

```
    "fileName": "550e8400-e29b-41d4-a716-446655440000.jpg",
    "originalFileName": "my-photo.jpg",
    "objectKey": "profile/550e8400-e29b-41d4-a716-446655440000.jpg",
    "directory": "PROFILE",
    "contentType": "image/jpeg",
    "fileSize": 245678,
    "fileSizeFormatted": "239.9 KB",
    "permanentUrl": "https://files.fursahub.com/fursa-userid/profile/550e8400.jpg",
    "thumbnailUrl": "https://files.fursahub.com/fursa-userid/profile/550e8400.jpg",
    "blurHash": "LKO2?U%2Tw=w]~RBVZRi};RPxuwH",
    "fileExtension": ".jpg",
    "fileType": "IMAGE",
    "isImage": true,
    "isVideo": false,
    "isDocument": false,
    "width": 1920,
    "height": 1080,
    "dimensions": "1920x1080",
    "checksum": "d41d8cd98f00b204e9800998ecf8427e",
    "uploadedAt": "2025-01-05T10:30:45",
    "uploadedBy": "550e8400-e29b-41d4-a716-446655440000",
    "isPublic": true
  }
}
```


Success Response Fields:

Field	Description
fileName	Generated unique filename
originalFileName	Original uploaded filename
objectKey	Full path in storage (directory/filename)
directory	Storage directory
permanentUrl	Public URL to access the file
thumbnailUrl	Thumbnail URL (same as permanentUrl for images)
blurHash	BlurHash string for image placeholders (null for non-images)
fileType	`IMAGE`, `VIDEO`, `DOCUMENT`, or `OTHER`
width, height	Image dimensions (null for non-images)

Field	Description
checksum	MD5 checksum for integrity verification

Error Responses:

File Too Large (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "File size exceeds maximum limit of 25MB",
  "action_time": "2025-01-05T10:30:45",
  "data": "File size exceeds maximum limit of 25MB"
}
```

Empty File (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "File is empty",
  "action_time": "2025-01-05T10:30:45",
  "data": "File is empty"
}
```

---

2. Upload Multiple Files

Purpose: Upload multiple files at once to the same directory.

Endpoint: POST `{base_url}/files/upload`

Access Level: □□ Protected

Authentication: Bearer Token

Content-Type: `multipart/form-data`

Request Parameters:

Parameter	Type	Required	Description	Validation

files	file[]	Yes	Array of files	Max 25MB each
directory	string	Yes	Target directory	enum values

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Files uploaded successfully",
  "action_time": "2025-01-05T10:35:00",
  "data": {
    "uploadedFiles": [
      {
        "fileName": "uuid1.jpg",
        "originalFileName": "photo1.jpg",
        "permanentUrl": "https://files.fursahub.com/bucket/path/uuid1.jpg",
        "blurHash": "LKO2?U%2Tw=w]...",
        "isImage": true
      },
      {
        "fileName": "uuid2.jpg",
        "originalFileName": "photo2.jpg",
        "permanentUrl": "https://files.fursahub.com/bucket/path/uuid2.jpg",
        "blurHash": "LAB2?Q%1Tu=x]...",
        "isImage": true
      }
    ],
    "totalFiles": 2,
    "successfulUploads": 2,
    "failedUploads": 0,
    "totalSize": 512000,
    "totalSizeFormatted": "500.0 KB",
    "uploadedAt": "2025-01-05T10:35:00",
    "message": "2 files uploaded successfully",
    "errors": null
  }
}
```


Partial Success Response (some files failed):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Files uploaded successfully",
  "action_time": "2025-01-05T10:35:00",
  "data": {
    "uploadedFiles": [...],
    "totalFiles": 3,
    "successfulUploads": 2,
    "failedUploads": 1,
    "message": "2 files uploaded successfully, 1 failed",
    "errors": [
      "Failed to upload large-file.zip: File size exceeds maximum limit of 25MB"
    ]
  }
}
```

BlurHash Usage

BlurHash is a compact representation of an image placeholder. Use it to show a blurred preview while the actual image loads.

Example BlurHash: `LKO2?U%2Tw=w]~RBVZRi};RPxuwH`

Frontend Implementation:

```
// React example with blurhash library
import { Blurhash } from "react-blurhash";

function ImageWithPlaceholder({ imageUrl, blurHash, width, height }) {
  const [loaded, setLoaded] = useState(false);

  return (
    <div style={{ position: 'relative' }}>
      {!loaded && blurHash && (
        <Blurhash
          hash={blurHash}
          width={width}
          height={height}
```

```
          resolutionX={32}
          resolutionY={32}
        />
      )}
      <img
        src={imageUrl}
        onLoad={() => setLoaded(true)}
        style={{ display: loaded ? 'block' : 'none' }}
      />
    </div>
  );
}
```

Frontend Implementation Guide

Profile Photo Upload

```
1. User selects photo
2. POST /files/upload-single
   ├── file: selected image
   └── directory: "PROFILE"
3. Get permanentUrl from response
4. POST /profile/photo with photoUrl
5. Display image with blurHash placeholder
```

Social Post with Images

```
1. User creates post, selects images
2. For each image: POST /files/upload-single
   ├── file: image
   └── directory: "SOCIAL_POST"
3. Collect all permanentUrls
4. Create post with image URLs array
5. Store blurHash for each image for feed display
```

File Upload Component

```
function uploadFile(file, directory) {
  const formData = new FormData();
  formData.append('file', file);
  formData.append('directory', directory);

  return fetch('{base_url}/files/upload-single', {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${accessToken}`
    },
    body: formData
  });
}
```

Validation Before Upload

```
const MAX_SIZE = 25 * 1024 * 1024; // 25MB
const ALLOWED_IMAGE_TYPES = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];

function validateFile(file) {
  if (file.size > MAX_SIZE) {
    throw new Error('File too large (max 25MB)');
  }
  if (!ALLOWED_IMAGE_TYPES.includes(file.type)) {
    throw new Error('Invalid file type');
  }
  return true;
}
```

Revision #4
Created 2 January 2026 08:07:38 by Admin Qbit
Updated 5 January 2026 16:22:30 by Admin Qbit