

MB-03-TS01. Technical Specification — MoodBoss API — Direct Upload to Cloudflare R2)

1. Basis and Objective

To implement a secure file upload mechanism allowing clients to upload files directly to Cloudflare R2 using the Direct Upload model (pre-signed PUT URL), including:

- generation of signed URLs by the server,
- upload confirmation (confirm endpoint),
- validation (size, MIME, SHA256),
- upload stage logging,
- initiation of asynchronous processing,
- basic lifecycle storage logic,
- storage of metadata in the database.

All server-side logic shall be implemented within the calculations container (MoodBoss Django application).

2. Solution Architecture

2.1 General Flow

Client → POST /files/presign

Server (calculations) → generates pre-signed PUT URL for R2

Client → uploads file directly to R2

Client → POST /files/confirm

Server:

- performs HEAD request to R2
- validates size / MIME / checksum
- updates status
- enqueues background task (if Celery is used)

Asynchronous processing

Status update

3. Cloudflare R2 — Requirements

S3-compatible R2 API is used:

Endpoint:

https://<ACCOUNT_ID>.r2.cloudflarestorage.com

Signature: AWS Signature v4

Upload method: PUT

Bucket: configured via environment variables

3.1 Environment Variables (via .env only, not Dockerfile)

The calculations container must use:

```
R2_ACCOUNT_ID=  
R2_ACCESS_KEY_ID=  
R2_SECRET_ACCESS_KEY=  
R2_BUCKET_NAME=  
R2_ENDPOINT=https://<ACCOUNT_ID>.r2.cloudflarestorage.com  
R2_PRESIGN_TTL=900  
R2_MAX_FILE_SIZE=10485760  
R2_ALLOWED_MIME=image/jpeg,image/png,application/pdf  
Storing keys in code or Dockerfile is strictly prohibited.
```

4. Functional Scope

4.1 Pre-Signed URL Generation Endpoint

POST

/calculations/api/v2/files/presign

Input:

```
{  
  "mime": "image/jpeg",  
  "size": 245678,  
  "checksum_sha256": "hex_string",  
  "profile_id": 426  
}
```

Logic:

- JWT authorization validation
- Allowed MIME validation
- Size check \leq R2_MAX_FILE_SIZE
- Generation of:
 - file_id (UUID)
 - object_key (e.g., uploads/{user_id}/{file_id})
- Generation of pre-signed PUT URL
- Creation of DB record with status presigned

Response:

```
{  
  "file_id": "uuid",  
  "object_key": "uploads/123/uuid",  
  "upload_url": "https://...",  
  "expires_in": 900  
}
```

4.2 Confirm Endpoint

POST

/calculations/api/v2/files/confirm

Input:

```
{
  "file_id": "uuid",
  "object_key": "uploads/123/uuid",
  "size": 245678,
  "mime": "image/jpeg",
  "checksum_sha256": "hex_string"
}
```

Logic:

- Verification that file_id belongs to the user
- Validation of current status
- HEAD request to R2:
 - verification of actual file size
- MIME validation
- Checksum validation:
 - either via object metadata,
 - or via server-side recalculation (if required)
- Status update → confirmed
- Enqueue Celery task
- Logging of stages

Response:

```
{
  "file_id": "uuid",
  "status": "confirmed"
}
```

5. Validations

5.1 File Size

Exceeding limit → 400 file_size_exceeded

5.2 MIME

Unsupported type → 400 invalid_mime_type

5.3 SHA256

Mismatch → 422 checksum_mismatch

6. Logging (Mandatory)

The following events must be logged:

- presign_issued
- confirm_requested
- validation_passed
- validation_failed
- async_processing_enqueued
- processing_started
- processing_completed
- processing_failed

Each log entry must contain:

- file_id
- user_id
- profile_id
- timestamp
- error_code (if applicable)

7. Database Model

Table example: UploadedFileModel

Mandatory fields:

- id (UUID)
- user_id
- profile_id (nullable)
- object_key
- bucket
- checksum_sha256
- size
- mime
- status
- error_code
- error_message
- created_at
- confirmed_at
- processed_at

8. Statuses

- created
- presigned
- confirmed
- processing
- processed
- failed
- expired

9. Asynchronous Processing

After confirmed:

- task is enqueued in Celery (if used in calculations),
- task downloads object from R2,
- performs required business processing,
- updates status.

10. Lifecycle Logic

Within the calculations container:

- cleanup of records in presigned status if confirm was not called within N hours;
- optional deletion of object from R2 if status is failed;
- CRON / periodic task (Celery Beat) to check expired records.

11. Security

- JWT authorization is mandatory.
- Confirmation of another user's file_id is prohibited.
- Pre-signed URL must have TTL.
- object_key is generated server-side and cannot be arbitrarily modified by the client.

12. Acceptance Criteria

The work shall be considered completed if:

- Pre-signed URL successfully uploads file to Cloudflare R2.
- Confirm endpoint validates:
 - size,
 - MIME,
 - checksum.
- Successful upload triggers asynchronous processing.
- Status transitions are correct.
- Full traceability by file_id is visible in logs.
- All R2 keys are read exclusively from .env.
- No hard dependency on AWS; S3-compatible R2 interface is used.

13. Signatures

Contractor

Vistadi LLC

Director:  V. Dildin

Customer

ONERY OVERSEAS LIMITED

Director: _____ A. Boulitsidou