RcertVault Digital Signature API

A token-based API for secure digital signing of PDFs and text documents.

Base URL
`http://202.51.70.18/api/v1/`

All endpoints except `/redoc/` require authentication.

Authentication

Each request must include an **Authorization** header with a token to authenticate the user and enforce permissions.

Using tokens

Tokens are secrets bound to user accounts. Include them as:
Authorization: Token
text- The prefix is exactly `Token` followed by a space.

- Do not use other schemes like `Bearer` or `Basic`.
- Never expose tokens in client-side code or version control.
- Rotate tokens regularly for security.

> **Note:** Tokens inherit the permissions of the associated user account.

Common authentication errors

If authentication fails, you'll receive a **403 Forbidden** response. Common causes include:

Cause	Fix
Header missing	Add the `Authorization` header.
Incorrect prefix	Use `Token` exactly (note the space).
Invalid or expired token	Generate a new token or check permissions.
Insufficient permissions	Verify the user's role and access.

Signing Flow Overview

The signing process establishes a session for secure operations. There are two paths based on your account configuration:

1. **Auto-signing enabled**: Retrieve or create a session directly.
2. **Manual signing**: Complete a challenge to verify your signing key.

Sessions expire after **24 hours** or when the PDF limit is reached. Only one active session per user, unless forced.

> **Note:** Use `/signing-challenge/` as the entry point.

Endpoints

Obtaining a signing session or challenge

POST `/signing-challenge/`

This endpoint initiates the workflow by returning an active session (if available) or generating a challenge for manual verification.

Request example

```
{
  "force_create": false  // Optional, default: false
}

Set force_create to true to invalidate any existing session and start fresh.
If auto-signing is enabled, an existing session is reused unless forced.

Responses
For auto-signing with an active session:
JSON{
  "signing_session_id": "uuid-string",
  "max_pdfs": 10,
  "expires_in_sec": 50884,
  "message": "Existing active signing session returned"
}
For manual signing:
JSON{
  "challenge_id": "uuid-string",
  "challenge_text": "base64-challenge-string",
  "expires_in_sec": 300
}
Sign the challenge_text (as bytes) using an external tool like DsignerClient or
EMSigner, then verify it within 5 minutes.

Errors:
400 Bad Request: Malformed JSON or invalid fields.
403 Forbidden: Authentication failed.

Verifying the challenge
POST /verify-challenge/
Required for manual signing. Submits the signed challenge to establish a session.
Request example
JSON{
    "force_create": false , // Optional, default: false
   "challenge_id": "uuid-string",
   "signature": "base64-or-raw-sig",
}
Response
JSON{
```

```
  "signing_session_id": "uuid-string",
  "max_pdfs": 10,
  "expires_in_sec": 86400,
  "message": "Signing session established successfully"
}
```

The challenge is single-use and deleted after verification to prevent replays.

Errors

400 Bad Request: Missing fields.
403 Forbidden: Authentication failed.
404 Not Found: Challenge expired or not found.

Verification failed:

```
JSON{
  "error": "Invalid or expired challenge"
}
```

Signing PDFs in bulk

POST /bulk-sign/

Signs multiple PDFs using an active session. Supports up to the max_pdfs limit (typically 10).

Request example

```
JSON{
  "signing_session_id": "uuid-string",
  "signature_box": "Coordinates as 'x1,y1,x2,y2' Eg: 400,100,600,200",
  "signature_page": "default: all, Page to show sign (int as string or 'all')",
  "location": "default: Nepal, maxLength: 255, Location text to put in signature",
  "apply_stamp": "default: false, If True, user must upload a stamp image; if False, user provides signature text",
  "signature_stamp": "Stamp image file (required if apply_stamp=true)"
}
```

signature_coordinates is optional for visual stamps.

Coordinates use PDF points (bottom-left origin).

Response

```
JSON{
  "status": "success",
  "signed_pdfs": [
    {
      "status": "signed",
      "signature": "Hashed data",
      "time_sec": 0.81
    }
  ],
  "signed_count": 1,
  "message": "Signing completed"
}
```

Errors

400 Bad Request: Invalid coordinates (e.g., x2<= x1).
403 Forbidden: Session mismatch or authentication failed.
404 Not Found: Session not found.
429 Too Many Requests: Exceeded PDF limit.

```
Verifying signatures
POST /verify/verify_text/
Verifies signatures on text, forms, or PDFs. Supports JSON or multipart/form-data.
Request examples
For JSON:
JSON{
  "text": "string",
  "signed_text": "string"
}
For multipart:
textpdf=@signed-document.pdf
Response
Always 200 OK if processed:
JSON{
  "status": "success",
  "result": [
    [
      {
        "cert_name": "Test1",
        "signature_ok": true,
        "hash_ok": true,
        "cert_ok": true
      }
    ],
    true
  ]
}

Supports multiple signatures.
Final boolean: All checks passed.

Field details:

FieldMeaningTrue if...signature_okCryptographic validity and no alterationsValid
sig + no tamperinghash_okHash matches signed hashUnchanged documentcert_okTrusted,
valid, not revokedValid chain and dates
Errors (in response body)

No signatures:

JSON{
  "error": "Invalid Base64 signature"
}

Tampered: signature_ok and hash_ok false.
Expired cert: cert_ok false.

PDF Coordinate System
For visual signatures:

Origin at bottom-left (0,0).
X right, Y up; units in points (1/72 inch).
Define box from (x1, y1) to (x2, y2), with x2 > x1 and y2 > y1.
```

```
Examples for A4 (595x842 points):

Bottom-right: {"page":1, "x1":400, "y1":50, "x2":550, "y2":150}
Top-right: {"page":1, "x1":400, "y1":750, "x2":550, "y2":850}
Center: {"page":1, "x1":200, "y1":400, "x2":400, "y2":500}

Invalid coordinates return 400 Bad Request.
Security and Best Practices

Use HTTPS in production (upgrade from HTTP).
Include visual signatures for auditability.
Verify after signing.
Handle session expiry in your code.

Warning: Treat tokens as secrets and rotate them.
For testing and schema, visit /redoc/.
```