Apidocumentation

> **Source File:** generated-documents\technical-design\apidocumentation.md
> **Generated:** 15/07/2025 at 11:40:46
> **Generated by:** Requirements Gathering Agent - PDF Converter

APIDocumentation

Generated by adpa-enterprise-framework-automation v3.2.0
Category: technical-design
Generated: 2025-07-14T21:03:04.149Z
Description:

ADPA (Advanced Document Processing & Automation Framework) API Documentation

Version: 3.2.0
Base URL: `http://localhost:3001/api/v1`
License: MIT

---

1. API Overview

ADPA is an enterprise-grade, modular automation framework offering RESTful services for AI-powered document generation, project

management, and business analysis. The API is designed with an API-first approach, providing endpoints for generating documentation, managing templates, publishing to collaboration platforms, and integrating with AI and document services. It is built using Node.js, TypeScript, and Express.js, and is standards-compliant with OpenAPI 3.0.

---

2. Authentication Methods

The API supports the following authentication mechanisms:

- **API Key**: Pass via `X-API-Key` header.
- **JWT (Bearer Token)**: Pass via `Authorization: Bearer <token>` header.
- **OAuth2**: Used for integrations (e.g., SharePoint, Confluence); user flow as per provider.

Example (API Key):

```
X-API-Key: your-api-key-here
```

Example (JWT):

```
Authorization: Bearer eyJhbGciOi...
```

Most endpoints require authentication.

---

3. Endpoint Definitions

Health & Status

GET `/health`

Returns API health, uptime, version, and system metrics.

Response:

```
{
  "status": "healthy",
  "timestamp": "2025-06-22T13:30:00.000Z",
  "version": "3.2.0",
  "environment": "production",
  "uptime": 245.6,
  "memory": {"used": 31, "total": 32, "external": 3},
  "node": "v18.20.2"
}
```

GET `/health/ready`

Readiness probe for load balancers and orchestrators.

Response:

```
{ "ready": true, "timestamp": "2025-06-22T13:30:00.000Z" }
```

Document Generation

POST `/generate`

Generate a standards-compliant document.

Request (JSON):

```
{
  "templateId": "ca8d4758-03c5-4110-84a7-2f5bcd318539",
  "input": {
    "projectName": "API Modernization Initiative",
    "stakeholders": [ "PMO", "QA Lead" ],
    "requirements": [ ... ]
  },
  "outputFormat": "pdf",
```

```
  "aiProvider": "openai"
}
```

- `templateId` : string (required)
- `input` : object (required) – dynamic fields depend on template
- `outputFormat` : `"pdf"` | `"markdown"` | `"docx"` | `"json"`
- `aiProvider` : `"openai"` | `"google"` | `"github"` | `"ollama"` | `"azure"`

Response (JSON):

```
{
  "jobId": "f7c4abce-6e91-4d58-bfa3-9e1e2f1a2b0d",
  "status": "queued",
  "downloadUrl": "/documents/jobs/f7c4abce-6e91-4d58-bfa3-9e1e2f1a2b0d
  "outputFormat": "pdf"
}
```

GET `/documents/jobs`

List document generation jobs (paginated).

Query Parameters:

- `page` : number (default: 1)
- `limit` : number (default: 20)

Response:

```
{
  "total": 1,
  "page": 1,
  "limit": 20,
  "jobs": [
    {
      "jobId": "f7c4abce-6e91-4d58-bfa3-9e1e2f1a2b0d",
      "status": "completed",
      "outputFormat": "pdf",
```

```
      "createdAt": "2025-06-22T13:32:00.000Z"
    }
  ]
}
```

POST `/documents/convert`

Convert an uploaded document to another format.

Request:

- Multipart/form-data:
  - `file`: File (required)
  - `targetFormat`: "pdf" | "docx" | "markdown" | "json"
  - `metadata` (optional): JSON string

Response:

```
{
  "jobId": "c8e7c7d3-1ea3-4ee6-9cf0-4ee4c2f0b2c2",
  "status": "processing",
  "downloadUrl": "/documents/jobs/c8e7c7d3-1ea3-4ee6-9cf0-4ee4c2f0b2c2
}
```

Templates

GET `/templates`

List available document templates.

Query Parameters:

- `category`: string (optional)
- `framework`: "babok" | "pmbok" | "dmbok" (optional)
- `page`, `limit`: Pagination

Response:

```
{
  "total": 2,
  "templates": [
    {
      "templateId": "ca8d4758-03c5-4110-84a7-2f5bcd318539",
      "name": "BABOK v3 Requirements Elicitation",
      "framework": "babok",
      "category": "requirements-elicitation",
      "description": "Comprehensive BABOK v3 template...",
      "outputFormats": ["pdf", "markdown", "docx"]
    }
  ]
}
```


POST `/templates`

Create a new template.

Request (JSON):


```
{
  "name": "PMBOK Project Charter",
  "framework": "pmbok",
  "category": "project-charter",
  "description": "PMBOK 7th Edition compliant project charter template
  "templateData": { ... }
}
```


Response:


```
{
  "templateId": "e6f8c8e2-81b4-4d31-8c4b-4fa758c5e320",
  "status": "created"
}
```


GET `/templates/{id}`

Retrieve template details.

Response:

```
{
  "templateId": "ca8d4758-03c5-4110-84a7-2f5bcd318539",
  "name": "BABOK v3 Requirements Elicitation",
  "framework": "babok",
  "category": "requirements-elicitation",
  "description": "...",
  "templateData": { ... }
}
```

Integration: Confluence

POST `/confluence/publish`

Publish a document to Atlassian Confluence.

Request (JSON):

```
{
  "documentId": "f7c4abce-6e91-4d58-bfa3-9e1e2f1a2b0d",
  "spaceKey": "PROJ",
  "title": "API Modernization Charter"
}
```

Response:

```
{
  "status": "success",
  "confluenceUrl": "https://yourcompany.atlassian.net/wiki/spaces/PROJ
}
```

Integration: SharePoint

POST `/sharepoint/upload`

Upload a document to SharePoint Online.

Request (JSON):

```
{
  "documentId": "f7c4abce-6e91-4d58-bfa3-9e1e2f1a2b0d",
  "siteUrl": "https://company.sharepoint.com/sites/ProjectX",
  "folder": "/Shared Documents/Project Charters"
}
```

Response:

```
{
  "status": "success",
  "sharepointUrl": "https://company.sharepoint.com/sites/ProjectX/Shar
}
```

Frameworks

GET `/frameworks`

List all supported standards and frameworks.

Response:

```
{
  "frameworks": [
    { "name": "BABOK v3", "status": "production" },
    { "name": "PMBOK 7th Edition", "status": "production" },
    { "name": "DMBOK 2.0", "status": "beta" }
  ]
}
```

---

4. Request/Response Formats

- **Content-Type:** `application/json` (unless file upload)
- **File Uploads:** `multipart/form-data`
- **All responses are JSON unless otherwise specified.**
- **Dates:** ISO 8601 format

---

5. Error Codes

HTTP Status	Code	Description	Example
400	`bad_request`	Validation error, malformed request	Missing required field
401	`unauthorized`	Authentication required or invalid credentials	Missing/invalid API key or token
403	`forbidden`	Insufficient permissions	User lacks required role
404	`not_found`	Resource does not exist	Template not found
409	`conflict`	Resource conflict	Duplicate template name
413	`payload_too_large`	Uploaded file too large	File exceeds upload limit

HTTP Status	Code	Description	Example
429	`rate_limit_exceeded`	Too many requests	Exceeded API rate limit
500	`internal_error`	Unhandled server error	Unexpected exception
503	`service_unavailable`	Down for maintenance or overloaded	Maintenance window

Error Response Format:

```
{
  "error": {
    "code": "bad_request",
    "message": "Missing required field: templateId",
    "details": {}
  }
}
```

---

6. Rate Limiting

- **Default:** 60 requests/minute per API key or token
- **Headers Returned:**
  - `X-RateLimit-Limit`
  - `X-RateLimit-Remaining`
  - `Retry-After` (if limited)

Example Response Headers:

```
HTTP/1.1 429 Too Many Requests
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 0
Retry-After: 60
```

---

7. Versioning Strategy

- **Current Version Path:** `/api/v1/`
- Breaking changes will increment the major version (`/api/v2/`).
- All endpoints are backward compatible within a major version.
- Deprecated endpoints will be announced at least one minor version in advance.

---

8. Security Considerations

- **HTTPS is mandatory** in production deployments.
- **API keys, tokens, and OAuth credentials must be kept secret**.
- **Rate limiting** and **IP throttling** protect against abuse.
- **Input validation** and **output encoding** prevent injection attacks.
- **Helmet** middleware sets secure HTTP headers.
- **JWT** tokens should have reasonable expiry times; refresh regularly.
- **Audit logging** is enabled for all administrative and sensitive actions.
- **Integration endpoints** (Confluence/SharePoint) require OAuth2 flows and consent.
- **CORS** is strictly configured.

---

9. Example Usage

cURL: Generate a Document

```
curl -X POST https://api.company.com/api/v1/generate \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your-api-key" \
  -d '{
    "templateId": "ca8d4758-03c5-4110-84a7-2f5bcd318539",
    "input": { "projectName": "Modernization" },
    "outputFormat": "pdf"
  }'
```

Node.js: List Templates

```
import axios from 'axios';

const resp = await axios.get(
  'https://api.company.com/api/v1/templates',
  { headers: { 'X-API-Key': 'your-api-key' } }
);
console.log(resp.data.templates);
```

Python: Publish to Confluence

```
import requests

r = requests.post(
    'https://api.company.com/api/v1/confluence/publish',
    headers={'Authorization': 'Bearer <token>'},
    json={'documentId': '...', 'spaceKey': 'PROJ', 'title': 'API Chart
)
print(r.json())
```

---

10. Testing Guidelines

- **Run API health checks** before and after deployment (`/api/v1/health`, `/api/v1/health/ready`).

- **Test authentication scenarios**: valid/invalid/missing API key or JWT.
- **Validate all required fields** for POST/PUT endpoints.
- **Test all error codes**: 400, 401, 403, 404, 409, 429, 500, 503.
- **Test rate limiting** by exceeding requests per minute.
- **Test file uploads** with valid and oversized files.
- **Integration endpoints**: use sandbox/test environments for Confluence/SharePoint.
- **Check CORS policy** for browser-based clients.
- **Use provided Swagger/OpenAPI docs** at `/api-docs` for live endpoint testing.

---

11. API Documentation & Interactive Console

- **Swagger UI:** http://localhost:3001/api-docs
- **Redocly UI:** (if configured)
- **OpenAPI Spec:** `/api-specs/openapi.yaml`

---

12. Support

- GitHub Issues
- Documentation Wiki
- Enterprise Support

For full API schemas, integration guides, and live testing, see API Documentation Portal or the GitHub Wiki.
