Wbs Dictionary

Source File: generated-documents\planning\wbs-dictionary.md
Generated: 15/07/2025 at 11:37:33
Generated by: Requirements Gathering Agent - PDF Converter

WBS Dictionary

Generated by adpa-enterprise-framework-automation v3.2.0
Category: planning
Generated: 2025-07-14T21:22:02.975Z
Description: PMBOK WBS Dictionary with detailed descriptions

WBS Dictionary

Document Overview

Purpose and Scope:
This WBS Dictionary provides comprehensive definitions for each work package in the ADPA (Advanced Document Processing & Automation Framework) project. It clarifies the scope, deliverables, acceptance criteria, responsibilities, required competencies, estimates, dependencies, constraints, assumptions, and risks for every major component and subcomponent of the project.

Relationship to Work Breakdown Structure (WBS):
The WBS decomposes the ADPA project into manageable sections. This dictionary acts as the authoritative reference for all work packages, ensuring clarity in execution, accountability, and effective communication among project stakeholders.

How to Use This Document:
Project managers, team leads, developers, QA, and stakeholders should use this document to:

- Understand detailed work package requirements and expectations
- Assign and manage responsibilities
- Plan, estimate, and monitor progress
- Assess risks, dependencies, and constraints

Work Package Definitions

1.0 – Project Management & Standards Governance

Description:
Establish and manage the project's PMO and ensure all deliverables comply with BABOK v3, PMBOK 7, and DMBOK 2.0 standards. Oversee project planning, scheduling, risk, quality, and compliance activities.

Deliverables:

- Project Charter and Scope Documentation
- Stakeholder Management Plan
- Standards Compliance Matrix
- Risk and Quality Management Plans

Acceptance Criteria:

- Documentation reviewed and approved by stakeholders
- Compliance matrix aligns with BABOK, PMBOK, and DMBOK
- All risks and quality processes documented and tracked

Responsible Party: Project Manager / Standards Lead
Skills Required: PMP, CBAP, knowledge of BABOK/PMBOK/DMBOK, risk management
Estimated Effort: 80–100 hours
Estimated Duration: 4–6 weeks

Dependencies:

- **Predecessors:** Project initiation, resource allocation
- **Successors:** All downstream technical and integration work packages

Constraints:

- Adherence to regulatory and industry standards
- Availability of subject matter experts

Assumptions:

- Access to up-to-date standards documentation

Risks:

- Incomplete stakeholder requirements; mitigated by early engagement

---

2.0 – AI Processing Engine

Description:
Develop and orchestrate the AI engine supporting multi-provider (OpenAI, Google AI, Copilot, Ollama) document generation, context management, and intelligent workflows.

Deliverables:

- AI Provider Integration Modules
- Context Management System
- Provider Failover & Orchestration Logic

Acceptance Criteria:

- All listed providers supported with auto-failover
- Context injection functional and tested
- Minimum 98% uptime in test scenarios

Responsible Party: AI Engineering Team
Skills Required: TypeScript, Node.js, AI APIs, cloud services, context management
Estimated Effort: 120–160 hours
Estimated Duration: 6–8 weeks

Dependencies:

- **Predecessors:** Project management plans, API keys/configuration

- **Successors:** Document Generator, REST API, CLI modules

Constraints:

- Provider API rate limits
- Licensing for AI APIs

Assumptions:

- API access will be granted for all providers

Risks:

- Provider API changes; mitigated by modular abstraction

3.0 – Document Generator

Description:
Design and implement the template-based document generation engine, integrating industry standards and supporting Markdown, PDF, and premium Adobe Creative Suite outputs.

Deliverables:

- Document Generation Engine (Markdown, PDF, Adobe outputs)
- Template Library (BABOK, PMBOK, DMBOK)
- Template Selection and Content Analysis Modules

Acceptance Criteria:

- All templates pass unit/integration tests
- Output meets formatting and branding requirements
- Adobe Creative Suite phase outputs match design specifications

Responsible Party: Document Engineering Team
Skills Required: TypeScript, template engines, Adobe SDKs, Node.js, Puppeteer
Estimated Effort: 160–220 hours
Estimated Duration: 8–10 weeks

Dependencies:

- **Predecessors:** AI Processing Engine, Adobe SDK setup
- **Successors:** Integration Layer, REST API, CLI

Constraints:

- Adobe API access
- Template standardization

Assumptions:

- Corporate branding guidelines provided

Risks:

- Complexity of Adobe integration; mitigated by phased implementation

---

4.0 – REST API Server

Description:
Develop a production-grade REST API (Express.js), conforming to TypeSpec/OpenAPI 3.0, exposing all document

generation and integration features.

Deliverables:

- Express.js API Server
- OpenAPI 3.0/TypeSpec Documentation
- API Authentication and Authorization Modules

Acceptance Criteria:

- API passes all functional and security tests
- OpenAPI docs available and up-to-date
- Authentication mechanisms (API Key, JWT) functional

Responsible Party: Backend/API Team
Skills Required: Node.js, Express.js, TypeScript, security best practices, OpenAPI
Estimated Effort: 100–140 hours
Estimated Duration: 5–7 weeks

Dependencies:

- **Predecessors:** Document Generator, AI Engine
- **Successors:** CLI/Web/Admin Interface, Integration Layer

Constraints:

- Compliance with security and regulatory policies

Assumptions:

- Development and production infrastructure available

Risks:

- Security vulnerabilities; mitigated by code reviews and penetration testing

---

5.0 – CLI Interface

Description:
Build a robust CLI (Yargs-based) supporting all generation, integration, and management commands for end-users and automation scripts.

Deliverables:

- CLI Command Modules (document generation, integration, admin)
- Interactive AI Provider Selection Menu
- Help and Error Handling Features

Acceptance Criteria:

- All CLI commands function as documented
- Interactive menu meets UX requirements
- CLI supports batch and advanced workflows

Responsible Party: CLI/Tooling Team
Skills Required: TypeScript, Yargs, Node.js, UX for CLI
Estimated Effort: 60–80 hours
Estimated Duration: 4–5 weeks

Dependencies:

- **Predecessors:** REST API Server, Document Generator
- **Successors:** Integration Layer, Admin Interface

Constraints:

- Command-line usability
- Backward compatibility

Assumptions:

- Node.js runtime will be >=18.x

Risks:

- Breaking changes in core APIs; mitigated by regression testing

---

6.0 – Integration Layer

Description:
Implement and validate integrations with Adobe Document Services, Confluence, SharePoint, and version control (GitHub, GitLab).

Deliverables:

- Adobe Creative Suite Integration (InDesign, Illustrator, Photoshop APIs)
- Confluence and SharePoint Integration Modules
- VCS Integration Scripts and Automation

Acceptance Criteria:

- Each integration passes end-to-end test cases
- OAuth2 and enterprise security compliance for all integrations
- Integration documentation complete

Responsible Party: Integration Team
Skills Required: TypeScript, API integration, Adobe SDKs, Microsoft Graph, Atlassian APIs, OAuth2
Estimated Effort: 140–180 hours
Estimated Duration: 7–9 weeks

Dependencies:

- **Predecessors:** REST API, CLI, Document Generator
- **Successors:** Admin Interface, Deployment

Constraints:

- Third-party API rate and licensing limits

Assumptions:

- API keys and credentials will be provisioned

Risks:

- Changes in third-party APIs; mitigated by monitoring and modularization

---

7.0 – Admin Web Interface

Description:
Develop an enterprise-grade admin portal (Next.js, React) for managing configurations, templates, integrations, and

analytics.

Deliverables:

- Admin Portal (Next.js/React)
- User and Project Management UI
- Analytics & Reporting Dashboard

Acceptance Criteria:

- All portal features pass UAT
- Role-based access control implemented and tested
- Analytics dashboard meets metric specifications

Responsible Party: Frontend/Web Team
Skills Required: Next.js, React, Tailwind CSS, REST APIs, UX/UI
Estimated Effort: 120–160 hours
Estimated Duration: 6–8 weeks

Dependencies:

- **Predecessors:** REST API, Integration Layer
- **Successors:** Deployment, User Training

Constraints:

- Browser compatibility
- Security standards for enterprise portals

Assumptions:

- Design assets and branding provided

Risks:

- UI/UX design delays; mitigated by prototyping and feedback cycles

---

8.0 – Analytics & Reporting

Description:
Implement analytics collection, metrics dashboards, and reporting for usage, system health, and compliance.

Deliverables:

- Usage Metrics Collection Pipeline
- Health Monitoring/Alerting
- Reporting Dashboard and Export Features

Acceptance Criteria:

- Real-time metrics available
- All reports exportable in standard formats
- Health checks comply with enterprise standards

Responsible Party: Analytics/DevOps Team
Skills Required: Node.js, monitoring tools, dashboard frameworks
Estimated Effort: 60–90 hours
Estimated Duration: 4–5 weeks

Dependencies:

- **Predecessors:** REST API, Admin Interface
- **Successors:** None (supports ongoing operations)

Constraints:

- Data privacy/compliance (GDPR, etc.)

Assumptions:

- Metrics endpoints exposed in API

Risks:

- Misconfigured monitoring; mitigated by standard templates

9.0 – Testing & Quality Assurance

Description:
Design and execute comprehensive test suites for unit, integration, performance, and compliance testing across all modules.

Deliverables:

- Automated Unit and Integration Test Suites
- Performance and Load Test Scripts
- Compliance Test Reports

Acceptance Criteria:

- > *95% test coverage on critical modules*
- Performance meets defined benchmarks
- Compliance tests pass for all frameworks

Responsible Party: QA/Test Automation Team
Skills Required: Jest, TypeScript, CI/CD, test automation
Estimated Effort: 80–100 hours
Estimated Duration: 4–6 weeks

Dependencies:

- **Predecessors:** All functional components
- **Successors:** Deployment/Release

Constraints:

- Test data availability

Assumptions:

- Staging environment available

Risks:

- Last-minute regression issues; mitigated by continuous testing

---

10.0 – Deployment & DevOps

Description:
Automate build, deployment, and release pipelines, including packaging (npm, Docker), CI/CD, and infrastructure setup.

Deliverables:

- Build and Deployment Scripts (npm, Docker)
- CI/CD Pipelines (GitHub Actions)
- Release Management Documentation

Acceptance Criteria:

- All environments deployable via automation
- Docker images pass security scans
- Release notes published per version

Responsible Party: DevOps/Release Team
Skills Required: CI/CD, Docker, npm, cloud deployment
Estimated Effort: 60–80 hours
Estimated Duration: 3–5 weeks

Dependencies:

- **Predecessors:** All code complete, tests passing
- **Successors:** User Training, Operations

Constraints:

- Cloud resource quotas
- Security scan requirements

Assumptions:

- Access to deployment infrastructure

Risks:

- Pipeline misconfigurations; mitigated by peer review

11.0 – Documentation & Training

Description:
Produce comprehensive user, administrator, and developer documentation, plus deliver training sessions for enterprise users and technical staff.

Deliverables:

- User Guides (CLI, API, Admin Portal)
- Developer and Integration Documentation
- Training Materials and Recordings

Acceptance Criteria:

- Documentation reviewed and approved
- Training sessions conducted and feedback collected

Responsible Party: Documentation & Training Team
Skills Required: Technical writing, standards compliance, training delivery
Estimated Effort: 50–70 hours
Estimated Duration: 3–4 weeks

Dependencies:

- **Predecessors:** Feature completion, stable interfaces
- **Successors:** User onboarding, support

Constraints:

- Language and accessibility requirements

Assumptions:

- Access to SMEs for content review

Risks:

- Documentation lagging behind development; mitigated by documentation sprints

---

Work Package Cross-Reference

WBS Code	Name	Responsible Party	Key Deliverables	Dependencies
1.0	Project Management & Standards Governance	Project Manager	PMO docs, compliance matrix	Project start
2.0	AI Processing Engine	AI Engineering Team	AI integration modules, context mgmt	1.0
3.0	Document Generator	Document Engineering	Generation engine, template libs	2.0
4.0	REST API Server	Backend/API Team	API server, OpenAPI docs	3.0
5.0	CLI Interface	CLI/Tooling Team	CLI modules, interactive menu	4.0
6.0	Integration Layer	Integration Team	Integrations (Adobe, Confluence, etc.)	5.0
7.0	Admin Web Interface	Frontend/Web Team	Admin portal, reporting	6.0
8.0	Analytics & Reporting	Analytics/DevOps Team	Metrics pipeline, dashboards	7.0
9.0	Testing & Quality Assurance	QA/Test Automation Team	Test suites, compliance reports	8.0
10.0	Deployment & DevOps	DevOps/Release Team	CI/CD, Docker images, release docs	9.0
11.0	Documentation & Training	Documentation Team	Guides, training, API docs	10.0

Responsibility Matrix

Work Package	Project Mgmt	AI Eng	Doc Eng	Backend	CLI	Integration	Frontend	QA	DevOps	D
1.0 Project Mgmt	X									
2.0 AI Engine		X								
3.0 Doc Generator			X							
4.0 REST API				X						
5.0 CLI					X					
6.0 Integration						X				
7.0 Admin Interface							X			
8.0 Analytics									X	
9.0 Testing		X	X	X	X	X	X	X	X	
10.0 Deployment									X	
11.0 Documentation	X	X	X	X	X	X	X	X	X	X

Dependency Network Overview

- 1.0 → 2.0 → 3.0 → 4.0 → 5.0 → 6.0 → 7.0 → 8.0 → 9.0 → 10.0 → 11.0
- Integration and feedback loops occur at each major handoff
- Testing/QA and Documentation run concurrently with late-phase development

---

Dictionary Maintenance

Update Procedures:

- Updates made at each phase gate or major change in scope
- Only project leads or WBS owners may submit changes

Version Control:

- Managed in project repository (`docs/WBS-DICTIONARY.md`)
- Change log and version number updated with each revision

Review Schedule:

- Reviewed at project kickoff, after each major milestone, and quarterly during maintenance/support phases
- Ad hoc review for critical changes or risk events

This WBS Dictionary is a living document. All project staff are responsible for reviewing relevant work package definitions prior to beginning assigned tasks and for reporting discrepancies or required updates to project management.
