CDW Documentation

SOW Analyzer – Teams Bot Application

SOW Analyzer – Teams Bot Application

1. Purpose and Scope

This document provides **full technical documentation** for the SOW Analyzer Teams Bot application. It is intended for:

- Engineers maintaining or extending the system
- Security and architecture reviewers
- Operations and platform teams
- Knowledge transfer and onboarding

This document describes:

- Application runtime behavior
- Source code responsibilities (file-by-file)
- Authentication and cross-tenant identity
- Teams message and file upload handling
- Data handling, storage, and retention
- Error cases and operational constraints

—

2. Functional Description

The SOW Analyzer is a **document ingestion and analysis system** exposed through **Microsoft Teams**.

Users interact with the system by uploading a **PDF Statement of Work** directly into a **1:1 Teams chat** with a bot. The system:

1. Validates the uploaded file
2. Downloads the PDF securely
3. Compares the document against known SOW templates
4. Uses Azure OpenAI to generate a structured comparison
5. Returns a summarized analysis to the user

At no point does the system store uploaded SOWs long-term.

—

3. High-Level System Flow

3.1 Request Lifecycle

```
User uploads PDF in Teams
 → Teams stores file in OneDrive/SharePoint
 → Teams sends attachment metadata to bot
 → Bot validates + downloads file
 → Bot POSTs PDF to analyzer endpoint
 → Analyzer extracts text and compares templates
 → Azure OpenAI produces structured diff
 → Result returned to Teams
```

3.2 Key Constraints

- PDF-only uploads
- Personal chat only
- Stateless processing
- No persistence of uploaded documents

—

4. Tenant and Identity Model

4.1 Tenant Separation

Component	Tenant
———-	——–
Teams users	CDW Tenant
Bot App Registration	CDW Tenant
Azure Bot Service	Azure Tenant
Web App (FastAPI)	Azure Tenant
Azure OpenAI	Azure Tenant
Blob Storage (templates)	Azure Tenant

4.2 Bot Identity

The bot uses a **single-tenant App Registration in the CDW Tenant**.

This App Registration:

- Is referenced by Azure Bot Service
- Signs JWT tokens presented to the bot
- Is validated by the Bot Framework Adapter

Required environment variables:

```
MicrosoftAppId
MicrosoftAppPassword
```

```
MicrosoftAppTenantId
```

The bot **will reject** any token whose AppId does not match `MicrosoftAppId`.

—

5. Teams Integration Details

5.1 Teams App Manifest

Key configuration:

```
"bots": [
  {
    "botId": "<MicrosoftAppId>",
    "scopes": ["personal"],
    "supportsFiles": true
  }
]
```

5.2 Supported Interaction Types

Interaction	Supported
————	———–
1:1 Chat	YES
Group Chat	NO
Channel	NO
Adaptive Cards	Optional
Message Extensions	NO

5.3 File Upload Behavior

Teams uploads files to **OneDrive or SharePoint**, then sends metadata to the bot.

Expected attachment type:

```
application/vnd.microsoft.teams.file.download.info
```

The bot **does not receive raw bytes directly**.

—

6. Source Code Breakdown

6.1 app.py

Purpose

Acts as the FastAPI entry point for:

- Teams bot messages
- SOW analysis workflow

Endpoints

POST /api/messages

- Receives Teams activities
- Delegates processing to `teams_messages()`
- No business logic lives here

POST /analyze-sow

- Accepts multipart/form-data
- Expects field name: `file`
- Reads PDF bytes into memory
- Executes SOW analysis pipeline
- Returns structured JSON

Important Behavior

- No disk writes
- No blob uploads
- File exists only in memory

—

6.2 teams_bot.py

Purpose

Implements all Teams bot logic.

Major Responsibilities

- Bot Framework authentication

- Activity validation
- File validation and download
- Integration with analyzer
- User-facing messaging

Adapter Initialization

```
BotFrameworkAdapterSettings(
  MicrosoftAppId,
  MicrosoftAppPassword,
  channel_auth_tenant=MicrosoftAppTenantId
)
```

This enforces:

- Single-tenant authentication
- Cross-tenant correctness

on_turn() Execution Flow

1. Trust incoming service URL 2. Ignore non-message activities 3. Extract attachments 4. If no attachment → return silently 5. Extract filename + download URL 6. Enforce `.pdf` extension 7. Download file bytes 8. Validate PDF magic bytes (`%PDF`) 9. POST file to analyzer 10. Format and send result

PDF Enforcement

Two layers:

- Filename check
- Byte-level PDF header check

File Size Enforcement

Maximum allowed size enforced before analysis.

Storage Behavior

- File exists only as a Python byte array
- Eligible for garbage collection after request

—

6.3 gpt_compare_multi.py

Purpose

Performs AI-driven comparison between uploaded SOW and known templates.

Responsibilities

- Load templates
- Construct structured prompt
- Call Azure OpenAI
- Parse JSON response
- Normalize output fields

Output Schema

```
{
  "chosen_template_title": "string",
  "summary": "string",
  "missing_sections": [],
  "extra_sections": [],
  "changed_clauses": {}
}
```

AI Safety

- No fine-tuning
- No training on customer data
- Prompt-only inference

—

6.4 layout_client.py

Purpose

Provides access to stored SOW templates.

Storage

- Azure Blob Storage
- Read-only access

Important Notes

- Only templates are stored
- Uploaded SOWs are never written here

—

7. Data Handling and Retention

7.1 Uploaded PDFs

- Stored in memory only
- Lifetime limited to request scope
- Never written to disk or blob

7.2 External Storage

- Teams stores original file in OneDrive/SharePoint
- Retention governed by M365 policies

7.3 Logging

- File contents never logged
- Metadata only (filename, size, correlation id)

—

8. Error Handling

User-Facing Errors

Condition	Message
———	———
Non-PDF upload	“I can only analyze PDF files”
Download failure	“I couldn’t download the file”
Analyzer failure	“Analysis failed”

Internal Errors

- Logged without sensitive data
- Returned as generic user messages

—

9. Security Model

Authentication

- Bot Framework JWT validation
- AppId + TenantId enforcement

Authorization

- Teams-scoped access
- Optional user allow-listing via AAD object ID

Data Protection

- No persistent storage
- No request body logging
- No file caching

—

10. Operational Considerations

Deployment

- Azure App Service
- Environment variable configuration
- No startup jobs or migrations

Scaling

- Stateless
- Horizontal scale-out supported

Monitoring

- Application Insights optional
- Sensitive logging disabled

—

11. Known Limitations

- Personal chat only
- PDF files only
- Channel uploads not supported
- No user authentication beyond Teams identity

—

12. Compliance Summary

The system:

- Does not store customer documents
- Minimizes data exposure
- Uses Azure-native security boundaries
- Aligns with enterprise data governance practices

—

13. Maintenance Notes

Future enhancements may include:

- Channel support via Graph
- Adaptive Card responses
- Template versioning
- Per-user access controls