Bank Statement Processing Agent

## Complete Technical Documentation

**Autonomous AI-Powered Bank Statement Processing System**Azure Functions • OpenAI • Document Intelligence • BAI2 Conversion

Version: 1.0  
Generated: August 07, 2025  
Development Team: BankStatementAgent Project

## Overview

The BankStatementAgent is an autonomous AI-powered system deployed on Azure that automatically processes bank statement PDFs and converts them to BAI2 format for banking system integration. The agent operates continuously, monitoring for new files and processing them without human intervention.

### Key Features

- Autonomous Processing: Automatically detects and processes uploaded PDFs

- AI-Powered Extraction: Uses Azure Document Intelligence and OpenAI for intelligent data extraction

- Format Conversion: Converts bank statements to industry-standard BAI2 format

- Error Resilience: Handles various document formats and quality levels

- Complete Audit Trail: Comprehensive logging and file archival

- Real-time Monitoring: Multiple monitoring solutions for system oversight

### System Specifications

- Platform: Azure Functions (Python 3.10)

- Primary AI: OpenAI GPT-4.1 for transaction parsing

- Document AI: Azure Document Intelligence for OCR and structure detection

- Storage: Azure Blob Storage for file management

- Monitoring: Application Insights for logging and analytics

- Processing Capacity: Handles PDFs up to several MB, processes 50+ transactions per statement

---

## System Architecture

**Code (MERMAID):**

graph TB  
 subgraph "User Interface"  
 A[PDF Upload to Blob Storage]  
 B[Monitoring Scripts]  
 C[Azure Portal Dashboard]  
 end  
   
 subgraph "Azure Cloud Environment"  
 subgraph "BankStatementAgent Function App"  
 D[Blob Trigger Function]  
 E[Setup HTTP Function]  
 end  
   
 subgraph "AI Services"  
 F[Azure Document Intelligence]  
 G[OpenAI GPT-4.1]  
 end  
   
 subgraph "Storage"  
 H[Incoming Statements Container]  
 I[BAI2 Outputs Container]  
 J[Archive Container]  
 end  
   
 subgraph "Monitoring"  
 K[Application Insights]  
 L[Function Logs]  
 end  
 end  
   
 A --> H  
 H --> D  
 D --> F  
 D --> G  
 D --> I  
 D --> J  
 D --> K  
 B --> H  
 B --> I  
 B --> J  
 C --> K  
 E --> K

### Architecture Components

- Azure Function App: BankStatementAgent

- Resource Group: Azure\_AI\_RG

- Location: East US

- Runtime: Python 3.10

- Plan: Consumption (Serverless)

- Azure Document Intelligence: OCR and document structure analysis

- OpenAI Service: Transaction parsing and data interpretation

- Intelligent Processing: Adaptive document handling

- Storage Account: waazuse1aistorage

- Container: bank-reconciliation

- incoming-bank-statements/: Input PDFs

- bai2-outputs/: Generated BAI2 files

- archive/: Processed original PDFs

- Application Insights: BankStatementAgent-AppInsights

- Function Logs: Real-time execution monitoring

- Custom Scripts: PowerShell monitoring tools

---

## AI Agent Components

### Agent Characteristics

**Code (MERMAID):**

graph LR  
 subgraph "AI Agent Loop"  
 A[Perceive] --> B[Analyze]  
 B --> C[Reason]  
 C --> D[Act]  
 D --> E[Monitor]  
 E --> A  
 end  
   
 subgraph "Capabilities"  
 F[Autonomous Operation]  
 G[Adaptive Intelligence]  
 H[Goal-Oriented Behavior]  
 I[Error Recovery]  
 end  
   
 A -.-> F  
 B -.-> G  
 C -.-> H  
 D -.-> I

### Core Intelligence Components

- Blob Trigger: Automatically detects new PDF files

- File Validation: Checks file type and accessibility

- Queue Management: Handles processing queue efficiently

- Document Intelligence: Extracts text and structure using Azure AI

- Content Assessment: Evaluates document quality and type

- Fallback Strategies: Multiple extraction approaches for different document qualities

- OpenAI Integration: Uses GPT-4.1 for intelligent transaction parsing

- Context Understanding: Interprets banking terminology and formats

- Data Validation: Ensures extracted data integrity

- BAI2 Generation: Creates industry-standard banking format files

- File Management: Organizes outputs and archives originals

- Error Handling: Manages processing exceptions gracefully

---

## Data Flow

**Code (MERMAID):**

sequenceDiagram  
 participant U as User  
 participant BS as Blob Storage  
 participant AF as Azure Function  
 participant DI as Document Intelligence  
 participant OAI as OpenAI  
 participant BAI as BAI2 Generator  
 participant ARC as Archive  
 participant AI as App Insights  
   
 U->>BS: Upload PDF to incoming-bank-statements/  
 BS->>AF: Trigger blob event  
 AF->>DI: Send PDF for OCR analysis  
 DI->>AF: Return extracted text + structure  
 AF->>OAI: Send text for transaction parsing  
 OAI->>AF: Return structured transaction data  
 AF->>BAI: Generate BAI2 format file  
 BAI->>BS: Save to bai2-outputs/  
 AF->>ARC: Move original to archive/  
 AF->>AI: Log processing results  
 AF->>BS: Delete from incoming/

### Processing Steps Detail

**Code (PYTHON):**

# Blob trigger activates when file uploaded  
@app.blob\_trigger(arg\_name="myblob", path="bank-reconciliation/incoming-bank-statements/{name}")  
def process\_new\_file(myblob: func.InputStream):

**Code (PYTHON):**

# Azure Document Intelligence extraction  
result = document\_intelligence\_client.begin\_analyze\_document(  
 "prebuilt-layout",   
 myblob.read()  
)

**Code (PYTHON):**

# OpenAI transaction parsing  
response = openai\_client.chat.completions.create(  
 model="gpt-4.1",  
 messages=[{"role": "user", "content": parsing\_prompt}]  
)

**Code (PYTHON):**

# Generate banking format  
bai2\_content = convert\_to\_bai2(  
 transactions=parsed\_data,  
 account\_info=account\_details  
)

**Code (PYTHON):**

# Save results and archive original  
blob\_client.upload\_blob(bai2\_content)  
archive\_original\_file(original\_pdf)

---

## Deployment Guide

### Prerequisites

- Azure Subscription

- Azure CLI installed and configured

- Python 3.10+ environment

- Git repository access

### Deployment Steps

**Code (BASH):**

# Clone repository  
git clone <repository-url>  
cd Bank Statement Reconciliation  
  
# Create virtual environment  
python -m venv .venv  
.venv\Scripts\activate # Windows  
source .venv/bin/activate # Linux/Mac  
  
# Install dependencies  
pip install -r requirements.txt

**Code (BASH):**

# Login to Azure  
az login  
  
# Create resource group  
az group create --name Azure\_AI\_RG --location "East US"  
  
# Create storage account  
az storage account create \  
 --name waazuse1aistorage \  
 --resource-group Azure\_AI\_RG \  
 --location "East US" \  
 --sku Standard\_LRS  
  
# Create Function App  
az functionapp create \  
 --resource-group Azure\_AI\_RG \  
 --consumption-plan-location "East US" \  
 --runtime python \  
 --runtime-version 3.10 \  
 --functions-version 4 \  
 --name BankStatementAgent \  
 --storage-account waazuse1aistorage

**Code (BASH):**

# Create Document Intelligence  
az cognitiveservices account create \  
 --name az-use1-docintelligence-01 \  
 --resource-group Azure\_AI\_RG \  
 --kind FormRecognizer \  
 --sku S0 \  
 --location "East US"  
  
# Create OpenAI Service (or use existing)  
az cognitiveservices account create \  
 --name wrldopenai \  
 --resource-group Azure\_AI\_RG \  
 --kind OpenAI \  
 --sku S0 \  
 --location "East US"

**Code (BASH):**

# Deploy function code  
func azure functionapp publish BankStatementAgent --python  
  
# Configure environment variables  
az functionapp config appsettings set \  
 --name BankStatementAgent \  
 --resource-group Azure\_AI\_RG \  
 --settings \  
 "DOCINTELLIGENCE\_ENDPOINT=https://az-use1-docintelligence-01.cognitiveservices.azure.com/" \  
 "DOCINTELLIGENCE\_KEY=<your-key>" \  
 "AZURE\_OPENAI\_ENDPOINT=https://wrldopenai.openai.azure.com" \  
 "AZURE\_OPENAI\_KEY=<your-key>" \  
 "AZURE\_OPENAI\_DEPLOYMENT=gpt-4.1"

### Configuration Files

**Code (JSON):**

{  
 "IsEncrypted": false,  
 "Values": {  
 "AzureWebJobsStorage": "<storage-connection-string>",  
 "FUNCTIONS\_WORKER\_RUNTIME": "python",  
 "FUNCTIONS\_WORKER\_RUNTIME\_VERSION": "3.10",  
 "DOCINTELLIGENCE\_ENDPOINT": "<endpoint-url>",  
 "DOCINTELLIGENCE\_KEY": "<api-key>",  
 "AZURE\_OPENAI\_ENDPOINT": "<openai-endpoint>",  
 "AZURE\_OPENAI\_KEY": "<openai-key>",  
 "AZURE\_OPENAI\_DEPLOYMENT": "gpt-4.1"  
 }  
}

**Code (TXT):**

azure-functions  
azure-storage-blob  
azure-ai-documentintelligence  
openai  
requests  
tabulate

---

## Monitoring & Logging

### Monitoring Architecture

**Code (MERMAID):**

graph TB  
 subgraph "Monitoring Sources"  
 A[Function Execution Logs]  
 B[Application Insights]  
 C[Blob Storage Activity]  
 D[Custom Metrics]  
 end  
   
 subgraph "Monitoring Tools"  
 E[Azure Portal Dashboard]  
 F[PowerShell Scripts]  
 G[Real-time Monitor]  
 H[KQL Queries]  
 end  
   
 subgraph "Alert Systems"  
 I[Processing Failures]  
 J[Performance Degradation]  
 K[Resource Limits]  
 end  
   
 A --> E  
 B --> E  
 C --> F  
 D --> G  
 E --> I  
 F --> J  
 G --> K

### Monitoring Scripts

**Code (POWERSHELL):**

# Continuously monitors blob storage for new activity  
# Shows processing events as they happen  
# Alerts for stuck files or errors  
.\monitor\_realtime.ps1

**Code (POWERSHELL):**

# Shows historical processing statistics  
# Success rates and file counts  
# Performance metrics  
.\check\_processing\_log.ps1

**Code (POWERSHELL):**

# Comprehensive health checks  
# Error detection and diagnostics  
# Application Insights integration  
.\enhanced\_monitor.ps1

### Application Insights Queries

**Code (KQL):**

traces  
| where cloud\_RoleName == "BankStatementAgent"  
| where timestamp > ago(4h)  
| where message contains "PROCESSING" or message contains "BAI2"  
| order by timestamp desc  
| project timestamp, message, severityLevel

**Code (KQL):**

union traces, exceptions  
| where cloud\_RoleName == "BankStatementAgent"  
| where severityLevel >= 2  
| order by timestamp desc  
| project timestamp, message, severityLevel, operation\_Name

**Code (KQL):**

traces  
| where cloud\_RoleName == "BankStatementAgent"  
| where message contains "Duration="  
| extend duration = extract(@"Duration=(\d+)ms", 1, message)  
| where isnotempty(duration)  
| summarize avg(toint(duration)), max(toint(duration)), min(toint(duration)) by bin(timestamp, 1h)

### Key Metrics to Monitor

- Files processed per hour/day

- Average processing time

- Success/failure rates

- Queue depth (files waiting)

- Function execution duration

- Memory usage

- API response times

- Error rates

- Total transactions processed

- BAI2 file sizes

- Document types handled

- Processing accuracy

---

## API Reference

### Azure Function Endpoints

- Type: Blob Trigger

- Path: bank-reconciliation/incoming-bank-statements/{name}

- Method: Automatic (triggered by file upload)

Trigger Configuration:

**Code (JSON):**

{  
 "bindings": [  
 {  
 "name": "myblob",  
 "type": "blobTrigger",  
 "direction": "in",  
 "path": "bank-reconciliation/incoming-bank-statements/{name}",  
 "connection": "AzureWebJobsStorage"  
 }  
 ]  
}

- URL: https://bankstatementagent-e8f3ddc9bwgjfvar.eastus-01.azurewebsites.net/api/setup

- Method: GET

- Purpose: Initialize blob storage containers

Response Format:

**Code (JSON):**

{  
 "status": "success",  
 "containers\_created": [  
 "incoming-bank-statements",  
 "bai2-outputs",   
 "archive"  
 ],  
 "message": "Storage containers initialized successfully"  
}

### File Upload API

**Code (BASH):**

az storage blob upload \  
 --account-name "waazuse1aistorage" \  
 --container-name "bank-reconciliation" \  
 --name "incoming-bank-statements/statement.pdf" \  
 --file "local-statement.pdf" \  
 --account-key "<storage-key>"

**Code (HTTP):**

PUT https://waazuse1aistorage.blob.core.windows.net/bank-reconciliation/incoming-bank-statements/statement.pdf  
Authorization: SharedKey waazuse1aistorage:<signature>  
Content-Type: application/pdf  
Content-Length: <file-size>  
  
<PDF binary data>

### Processing Status API

**Code (BASH):**

# List BAI2 outputs  
az storage blob list \  
 --account-name "waazuse1aistorage" \  
 --container-name "bank-reconciliation" \  
 --prefix "bai2-outputs/" \  
 --output table  
  
# Check archived files  
az storage blob list \  
 --account-name "waazuse1aistorage" \  
 --container-name "bank-reconciliation" \  
 --prefix "archive/" \  
 --output table

---

## Troubleshooting

### Common Issues and Solutions

Symptoms:

- Files uploaded but not processed

- No BAI2 outputs generated

- Files remain in incoming folder

Diagnosis:

**Code (POWERSHELL):**

# Check function app status  
az functionapp show --name BankStatementAgent --resource-group Azure\_AI\_RG  
  
# Check for stuck files  
.\enhanced\_monitor.ps1

Solutions:

1. Verify function app is running

2. Check blob trigger configuration

3. Validate storage connection string

4. Review function logs for errors

Symptoms:

- Files processed but errors in logs

- Incomplete BAI2 files

- Missing archived files

Diagnosis:

**Code (KQL):**

exceptions  
| where cloud\_RoleName == "BankStatementAgent"  
| order by timestamp desc

Solutions:

1. Check AI service quotas and limits

2. Validate API keys and endpoints

3. Review document quality and format

4. Check network connectivity

Symptoms:

- Slow processing times

- Timeouts

- High resource usage

Diagnosis:

**Code (POWERSHELL):**

# Monitor processing times  
.\enhanced\_monitor.ps1  
  
# Check Azure metrics  
az monitor metrics list --resource <function-app-resource-id>

Solutions:

1. Optimize document processing

2. Increase function timeout limits

3. Scale function app plan

4. Implement caching strategies

### Error Codes and Messages

- 400 Bad Request: Invalid document format

- 429 Too Many Requests: Rate limit exceeded

- 500 Internal Server Error: Service unavailable

- 401 Unauthorized: Invalid API key

- 429 Rate Limited: Token limit exceeded

- 503 Service Unavailable: Model unavailable

- 404 Not Found: Container or blob doesn't exist

- 403 Forbidden: Access denied

- 409 Conflict: Blob already exists

### Diagnostic Commands

**Code (BASH):**

# Function app health check  
az functionapp show --name BankStatementAgent --resource-group Azure\_AI\_RG  
  
# List function app settings  
az functionapp config appsettings list --name BankStatementAgent --resource-group Azure\_AI\_RG  
  
# Check function logs  
az functionapp log tail --name BankStatementAgent --resource-group Azure\_AI\_RG  
  
# Storage account diagnostics  
az storage account show --name waazuse1aistorage --resource-group Azure\_AI\_RG

---

## Configuration

### Environment Variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Description** | **Required** | **Example** |
| `AzureWebJobsStorage` | Storage connection string | Yes | `DefaultEndpointsProtocol=https;AccountName=...` |
| `FUNCTIONS\_WORKER\_RUNTIME` | Runtime language | Yes | `python` |
| `FUNCTIONS\_WORKER\_RUNTIME\_VERSION` | Python version | Yes | `3.10` |
| `DOCINTELLIGENCE\_ENDPOINT` | Document Intelligence URL | Yes | `https://service.cognitiveservices.azure.com/` |
| `DOCINTELLIGENCE\_KEY` | Document Intelligence API key | Yes | `abcd1234...` |
| `AZURE\_OPENAI\_ENDPOINT` | OpenAI service URL | Yes | `https://service.openai.azure.com` |
| `AZURE\_OPENAI\_KEY` | OpenAI API key | Yes | `abcd1234...` |
| `AZURE\_OPENAI\_DEPLOYMENT` | OpenAI model deployment | Yes | `gpt-4.1` |

### Function Configuration

**Code (JSON):**

{  
 "version": "2.0",  
 "logging": {  
 "applicationInsights": {  
 "samplingSettings": {  
 "isEnabled": true,  
 "excludedTypes": "Request"  
 }  
 }  
 },  
 "extensionBundle": {  
 "id": "Microsoft.Azure.Functions.ExtensionBundle",  
 "version": "[4.\*, 5.0.0)"  
 },  
 "functionTimeout": "00:10:00"  
}

**Code (PYTHON):**

import azure.functions as func  
import logging  
  
app = func.FunctionApp()  
  
# Blob trigger configuration  
@app.blob\_trigger(  
 arg\_name="myblob",   
 path="bank-reconciliation/incoming-bank-statements/{name}",  
 connection="AzureWebJobsStorage"  
)  
def process\_new\_file(myblob: func.InputStream):  
 # Processing logic  
 pass  
  
# HTTP trigger configuration   
@app.route(route="setup", auth\_level=func.AuthLevel.ANONYMOUS)  
def setup\_containers(req: func.HttpRequest) -> func.HttpResponse:  
 # Setup logic  
 pass

### Performance Tuning

**Code (JSON):**

{  
 "functionTimeout": "00:10:00", // 10 minutes for large documents  
 "healthCheck": {  
 "delayBeforeFirstProbe": "00:00:30"  
 }  
}

**Code (JSON):**

{  
 "concurrency": {  
 "maxConcurrentRequests": 5,  
 "dynamicConcurrencyEnabled": true  
 }  
}

**Code (PYTHON):**

# In function\_app.py  
import gc  
  
def process\_large\_document(document\_data):  
 try:  
 # Process document  
 result = extract\_data(document\_data)  
 return result  
 finally:  
 # Force garbage collection for large files  
 gc.collect()

---

## Security

### Security Architecture

**Code (MERMAID):**

graph TB  
 subgraph "Security Layers"  
 A[Identity & Access Management]  
 B[Network Security]  
 C[Data Encryption]  
 D[API Security]  
 E[Audit & Compliance]  
 end  
   
 subgraph "Azure Security Services"  
 F[Azure AD]  
 G[Key Vault]  
 H[Private Endpoints]  
 I[Storage Encryption]  
 J[Activity Logs]  
 end  
   
 A --> F  
 B --> H  
 C --> I  
 D --> G  
 E --> J

### Access Control

- Authentication: Azure AD integration

- Authorization: Role-based access control (RBAC)

- Function Keys: HTTP trigger protection

- System Identity: Managed identity for Azure services

- Account Keys: Secure key management

- SAS Tokens: Limited-time access tokens

- Private Endpoints: Network isolation

- Encryption: At-rest and in-transit encryption

### Data Protection

- Storage: AES-256 encryption at rest

- Transit: TLS 1.2+ for all communications

- Keys: Azure Key Vault for key management

- Processed Files: Archived with retention policies

- Logs: 90-day retention in Application Insights

- BAI2 Files: Long-term storage with compliance requirements

### Compliance

- PCI DSS: Payment card data security

- SOX: Financial reporting compliance

- GDPR: Data privacy regulations

- Banking Secrecy Act: Anti-money laundering

- SOC 2 Type II: Security controls audit

- ISO 27001: Information security management

- FedRAMP: Government cloud security

- HIPAA: Healthcare data protection

### Security Best Practices

**Code (PYTHON):**

# Secure coding practices  
import os  
from azure.keyvault.secrets import SecretClient  
  
# Use Key Vault for sensitive data  
def get\_secure\_config(secret\_name):  
 vault\_url = os.environ["KEY\_VAULT\_URL"]  
 credential = DefaultAzureCredential()  
 client = SecretClient(vault\_url=vault\_url, credential=credential)  
 return client.get\_secret(secret\_name).value

**Code (BASH):**

# Secure deployment commands  
az functionapp config appsettings set \  
 --name BankStatementAgent \  
 --settings "@secure-settings.json" # Use file instead of command line  
  
# Enable system-assigned managed identity  
az functionapp identity assign \  
 --name BankStatementAgent \  
 --resource-group Azure\_AI\_RG

**Code (KQL):**

// Security monitoring query  
SecurityEvent  
| where EventID in (4625, 4648, 4672) // Failed logons, explicit credentials, special privileges  
| where TimeGenerated > ago(24h)  
| summarize count() by Account, Computer, EventID  
| order by count\_ desc

---

## Appendix

### File Formats

01,<sender-id>,<receiver-id>,<creation-date>,<creation-time>,<file-id>,<physical-record-length>,<block-size>,<version-number>,/  
02,<ultimate-receiver-id>,<originator-id>,<group-status>,<as-of-date>,<as-of-time>,<currency-code>,<as-of-date-modifier>,/  
03,<customer-account-number>,<currency-code>,<type-code>,<amount>,<item-count>,<funds-type>,<bank-reference>,<customer-reference>,/  
16,<type-code>,<amount>,<item-count>,<funds-type>,<availability>,<text>,/  
49,<account-control-total>,<number-of-records>,/  
98,<group-control-total>,<number-of-accounts>,<number-of-records>,/  
99,<file-control-total>,<number-of-groups>,<number-of-records>,/

- Digital PDFs: Text-based bank statements

- Scanned PDFs: Image-based documents (OCR processed)

- Multi-page: Statements spanning multiple pages

- Various Banks: Adaptable to different bank formats

- Size Limits: Up to 50MB per file

### Performance Benchmarks

|  |  |  |
| --- | --- | --- |
| **Metric** | **Typical Value** | **Maximum Tested** |
| \*\*Processing Time\*\* | 30-90 seconds | 5 minutes |
| \*\*File Size\*\* | 1-10 MB | 50 MB |
| \*\*Transactions\*\* | 10-100 per statement | 500+ |
| \*\*Accuracy\*\* | 95-99% | 99.5% |
| \*\*Throughput\*\* | 50 files/hour | 200 files/hour |

### Cost Estimation

- Function App: $0-50 (consumption based)

- Storage Account: $5-20 (based on volume)

- Document Intelligence: $100-500 (based on pages)

- OpenAI: $50-200 (based on tokens)

- Application Insights: $10-50 (based on logs)

Total Estimated Cost: $165-820/month (varies by usage)

### Support Contacts

- Azure Support: https://portal.azure.com/#blade/Microsoft\_Azure\_Support/HelpAndSupportBlade

- Documentation: https://docs.microsoft.com/azure/azure-functions/

- Community: https://stackoverflow.com/questions/tagged/azure-functions

- Azure Functions: Core compute platform

- Azure Storage: File management

- Document Intelligence: OCR and analysis

- OpenAI: Natural language processing

- Application Insights: Monitoring and logging

---

Document Version: 1.0

Last Updated: August 7, 2025

Created by: BankStatementAgent Development Team