To address the requirements for developing a component for information extraction from email messages, focusing on extracting invoice or purchase document details, we need a detailed solution document. This document will include technical steps, sample code snippets for proof of concept (POC), and a flow diagram to illustrate the process.

---

## \*\*Solution Document for Information Extraction from Email Messages\*\*

### \*\*1. Overview\*\*

The objective is to develop a component that extracts information related to invoices or purchase documents from emails in `.eml` or `.msg` formats. The component should classify file types, extract and analyze metadata, and process attachments to extract named entities. The final output should be a JSON object containing the relevant extracted information.

### \*\*2. Technical Flow\*\*

1. \*\*File Classification\*\*

2. \*\*Email Parsing and Metadata Extraction\*\*

3. \*\*Information Extraction\*\*

- Email body content

- Attached PDF documents

- Attached Word files (DOCX)

- Attached Excel files (XLSX)

- Password-protected PDFs

4. \*\*Generate JSON Output\*\*

5. \*\*Integration with FileNet\*\*

### \*\*3. Detailed Technical Steps\*\*

#### \*\*3.1. File Classification\*\*

\*\*Objective:\*\* Determine if the input file is `.eml` or `.msg`.

\*\*Steps:\*\*

1. Check file extension to classify the email type.

\*\*POC Code:\*\*

```python

import os

def classify\_file(file\_path):

\_, ext = os.path.splitext(file\_path)

if ext.lower() in ['.eml', '.msg']:

return ext.lower()

else:

raise ValueError("Unsupported file type")

# Example usage

file\_path = 'example.eml'

file\_type = classify\_file(file\_path)

print(f"File Type: {file\_type}")

```

#### \*\*3.2. Email Parsing and Metadata Extraction\*\*

\*\*Objective:\*\* Extract metadata and content type based on the email file type.

\*\*Steps:\*\*

1. For `.eml`, use the `email` library.

2. For `.msg`, use the `extract-msg` library.

\*\*POC Code:\*\*

```python

import email

from email import policy

from email.parser import BytesParser

import extract\_msg

def parse\_email(file\_path, file\_type):

if file\_type == '.eml':

with open(file\_path, 'rb') as f:

msg = BytesParser(policy=policy.default).parse(f)

metadata = {

'from': msg['from'],

'to': msg['to'],

'subject': msg['subject'],

'date': msg['date'],

'content\_type': msg.get\_content\_type()

}

return msg, metadata

elif file\_type == '.msg':

msg = extract\_msg.Message(file\_path)

metadata = {

'from': msg.sender,

'to': msg.to,

'subject': msg.subject,

'date': msg.date,

'content\_type': msg.message\_type

}

return msg, metadata

# Example usage

msg, metadata = parse\_email('example.eml', '.eml')

print(metadata)

```

#### \*\*3.3. Information Extraction\*\*

\*\*Objective:\*\* Extract named entities from the email body and attachments.

\*\*Steps:\*\*

1. \*\*Extract Email Body Content:\*\*

```python

def get\_body\_content(msg):

if msg.is\_multipart():

for part in msg.iter\_parts():

if part.get\_content\_type() == 'text/plain':

return part.get\_payload(decode=True).decode()

else:

return msg.get\_payload(decode=True).decode()

body\_content = get\_body\_content(msg)

print(body\_content)

```

2. \*\*Extract Attachments:\*\*

```python

from io import BytesIO

import PyPDF2

from docx import Document

import openpyxl

def extract\_attachments(msg):

attachments = {}

for part in msg.iter\_parts():

if part.get\_content\_disposition() == 'attachment':

filename = part.get\_filename()

content\_type = part.get\_content\_type()

data = part.get\_payload(decode=True)

if filename.endswith('.pdf'):

attachments['pdf'] = data

elif filename.endswith('.docx'):

attachments['docx'] = data

elif filename.endswith('.xlsx'):

attachments['xlsx'] = data

return attachments

attachments = extract\_attachments(msg)

```

3. \*\*Handle Password-Protected PDF:\*\*

```python

def read\_pdf\_with\_password(pdf\_data, password):

try:

pdf = PyPDF2.PdfReader(BytesIO(pdf\_data))

if pdf.is\_encrypted:

pdf.decrypt(password)

text = ""

for page in pdf.pages:

text += page.extract\_text()

return text

except Exception as e:

print(f"Error reading PDF: {e}")

# Example usage

pdf\_data = attachments.get('pdf')

if pdf\_data:

text = read\_pdf\_with\_password(pdf\_data, 'password123')

print(text)

```

4. \*\*Extract Named Entities:\*\*

For entity extraction, use an NLP library like `spaCy` for Named Entity Recognition (NER).

\*\*Install spaCy and download a model:\*\*

```bash

pip install spacy

python -m spacy download en\_core\_web\_sm

```

\*\*POC Code:\*\*

```python

import spacy

nlp = spacy.load("en\_core\_web\_sm")

def extract\_entities(text):

doc = nlp(text)

entities = {}

for ent in doc.ents:

if ent.label\_ in ['DATE', 'MONEY', 'ORG']:

entities[ent.label\_] = ent.text

return entities

# Example usage

email\_entities = extract\_entities(body\_content)

pdf\_text = read\_pdf\_with\_password(pdf\_data, 'password123') if pdf\_data else ''

pdf\_entities = extract\_entities(pdf\_text)

docx\_text = Document(BytesIO(attachments.get('docx'))).text if 'docx' in attachments else ''

docx\_entities = extract\_entities(docx\_text)

xlsx\_text = "Sample extracted text from XLSX" # Implement text extraction from XLSX if needed

xlsx\_entities = extract\_entities(xlsx\_text)

```

#### \*\*3.4. Generate JSON Output\*\*

\*\*Objective:\*\* Create a structured JSON output containing extracted information.

\*\*Steps:\*\*

```python

import json

def generate\_json\_output(metadata, email\_entities, pdf\_entities, docx\_entities, xlsx\_entities):

output = {

'from': metadata['from'],

'to': metadata['to'],

'subject': metadata['subject'],

'Invoice No.': email\_entities.get('INVOICE'),

'Invoice Date': email\_entities.get('DATE'),

'Amount': email\_entities.get('MONEY'),

'PDF Entities': pdf\_entities,

'DOCX Entities': docx\_entities,

'XLSX Entities': xlsx\_entities

}

return json.dumps(output, indent=4)

# Example usage

json\_output = generate\_json\_output(metadata, email\_entities, pdf\_entities, docx\_entities, xlsx\_entities)

print(json\_output)

```

#### \*\*3.5. Integration with FileNet\*\*

\*\*Objective:\*\* Handle file input/output operations with the FileNet system.

\*\*Steps:\*\*

1. Configure FileNet to send the email files to the component.

2. Implement file retrieval and processing using FileNet API or SDK.

\*\*POC Code:\*\*

```python

# Placeholder for FileNet integration

def process\_file\_from\_filenet(file\_path):

# Retrieve file from FileNet

# For demonstration, assume file\_path is received from FileNet

file\_type = classify\_file(file\_path)

msg, metadata = parse\_email(file\_path, file\_type)

body\_content = get\_body\_content(msg)

attachments = extract\_attachments(msg)

email\_entities = extract\_entities(body\_content)

pdf\_text = read\_pdf\_with\_password(attachments.get('pdf'), 'password123') if 'pdf' in attachments else ''

pdf\_entities = extract\_entities(pdf\_text)

docx\_text = Document(BytesIO(attachments.get('docx'))).text if 'docx' in attachments else ''

docx\_entities = extract\_entities(docx\_text)

xlsx\_text = "Sample extracted text from XLSX" # Implement text extraction from XLSX if needed

xlsx\_entities = extract\_entities(xlsx\_text)

json\_output = generate\_json\_output(metadata, email\_entities, pdf\_entities, docx\_entities, xlsx\_entities)

return json\_output

# Example usage

output = process\_file\_from\_filenet('example.eml')

print(output)

```

### \*\*4. Flow Diagram\*\*

Below is a high-level flow diagram illustrating the end-to-end process:

```

+------------------------+

| Receive Email |

| (from FileNet) |

+------------------------+

|

v

+------------------------+

| Classify File Type |

| (.eml or .msg) |

+------------------------+

|

v

+------------------------+

| Parse Email & Metadata |

+------------------------+

|

v

+------------------------+

| Extract Email Body |

| and Attachments |

+------------------------+

|

v

+------------------------+

| Extract Named Entities |

| from Text and Attachments |

+------------------------+

|

v

+------------------------+

| Generate JSON Output |

+------------------------+

|

v

+------------------------+

| Output JSON |

+------------------------+

```

### \*\*5. Conclusion\*\*

This solution document provides a detailed plan for developing a component that extracts information from email messages, focusing on invoices or purchase documents. It includes technical steps, code snippets for proof of concept, and a flow diagram to guide the implementation process. For a production environment

, ensure to handle exceptions, validate inputs, and integrate securely with the FileNet system.