Certainly! Below is a comprehensive solution document to address the requirements for developing a component for information extraction from email messages. This document covers technical steps, sample code snippets for proof of concept (POC), and a flow diagram to illustrate the end-to-end process.

---

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

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

The goal is to develop a component that extracts information from email messages, which can be in `.eml` or `.msg` formats. The system will classify the file type, parse email metadata and content, extract information from various attachments, and generate a JSON output. The component will receive emails through a FileNet system.

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

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

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

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

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 `email` library.

2. For `.msg`, use `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 = {

'subject': msg['subject'],

'from': msg['from'],

'to': msg['to'],

'date': msg['date'],

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

}

return msg, metadata

elif file\_type == '.msg':

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

metadata = {

'subject': msg.subject,

'from': msg.sender,

'to': msg.to,

'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 email body content, attachments (PDF, DOCX, XLSX), and handle password-protected PDFs.

\*\*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'):

if 'application/pdf' in content\_type:

attachments['pdf'] = data

elif filename.endswith('.docx'):

if 'application/vnd.openxmlformats-officedocument.wordprocessingml.document' in content\_type:

attachments['docx'] = data

elif filename.endswith('.xlsx'):

if 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet' in content\_type:

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)

```

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

\*\*Objective:\*\* Create a structured JSON output containing email metadata, body content, and extracted attachments.

\*\*Steps:\*\*

```python

import json

def generate\_json\_output(metadata, body\_content, attachments):

output = {

'metadata': metadata,

'body\_content': body\_content,

'attachments': {

'pdf': attachments.get('pdf', '').decode() if 'pdf' in attachments else None,

'docx': attachments.get('docx', '').decode() if 'docx' in attachments else None,

'xlsx': attachments.get('xlsx', '').decode() if 'xlsx' in attachments else None

}

}

return json.dumps(output, indent=4)

# Example usage

json\_output = generate\_json\_output(metadata, body\_content, attachments)

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 as per the 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)

json\_output = generate\_json\_output(metadata, body\_content, attachments)

return json\_output

# Example usage

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

print(output)

```

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

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

```

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

| Receive Email |

| (from FileNet) |

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

|

v

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

| Classify File Type |

| (.eml or .msg) |

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

|

v

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

| Parse Email & Metadata |

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

|

v

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

| Extract Body Content |

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

|

v

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

| Extract Attachments |

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

|

v

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

| Handle Password-Protected PDFs |

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

|

v

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

| Generate JSON Output |

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

|

v

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

| Output JSON |

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

```

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

This solution outlines the technical steps required to implement an information extraction component for email messages. It covers file classification, metadata extraction, content extraction, and JSON output generation. Additionally, integration with FileNet is considered to handle file input and output. The provided code snippets serve as a proof of concept for each requirement.

For a production environment, ensure to handle edge cases, implement robust error handling, and adhere to security best practices, especially for handling sensitive information and password-protected files.