Source: C:\Users\Admin\Desktop\CMC\ms1_emailIngestion\core\unified_email_processor.py

```
"""
Unified Email Processor
X■ lý email th■ng nh■t cho c■ polling và webhook
■■m b■o không duplicate, không b■ sót
"""
import json
import os
import base64
import requests
from typing import List, Dict, Optional
from datetime import datetime, timezone
from core.session_manager import session_manager
from utils.config import (
    ATTACH_DIR,
    SPAM_PATTERNS,
    MS2_CLASSIFIER_BASE_URL,
    MS4_PERSISTENCE_BASE_URL
)

class EmailProcessor:
    """Core processor x■ lý email"""

    def __init__(self, token: str):
        self.token = token
        self.headers = {"Authorization": f"Bearer {token}"}
        os.makedirs(ATTACH_DIR, exist_ok=True)

    def process_email(self, message: Dict, source: str = "unknown") -> bool:
        """X■ lý m■t email"""
        msg_id = message.get("id")
        if not msg_id:
            print(f"[EmailProcessor] Missing message ID")
            return False

        # Ki■m tra duplicate
        if session_manager.is_email_processed(msg_id):
            print(f"[EmailProcessor] [{source}] Email {msg_id} already processed")
            return False

        subject = message.get("subject", "")
        sender = message.get("from", {}).get("emailAddress", {}).get("address", "")
        received_at = message.get("receivedDateTime", "")

        print(f"[EmailProcessor] [{source}] Processing: {msg_id}")
        print(f"  Subject: {subject}")
        print(f"  From: {sender}")

        try:
            # Step 1: L■c spam
            if self._is_spam(sender):
                print(f"[EmailProcessor] SPAM detected, moving to junk")
                self._move_to_junk(msg_id)
                session_manager.register_processed_email(msg_id)
                return True

            # Step 2: L■u attachments
            self._save_attachments(msg_id)

            # Step 3: G■i metadata ■■n MS2 (Classifier)
            self._forward_to_classifier(message)

            # Step 4: G■i metadata ■■n MS4 (Persistence)
            self._forward_to_persistence(message)

            # ■■ng ký ■ã x■ lý
            session_manager.register_processed_email(msg_id)

            print(f"[EmailProcessor] [{source}] Successfully processed: {msg_id}")
            return True

        except Exception as e:
            print(f"[EmailProcessor] [{source}] Error processing {msg_id}: {e}")
            return False

    def batch_process_emails(self, messages: List[Dict], source: str = "polling") -> Dict:
        """X■ lý batch emails"""
        result = {
            "total": len(messages),
            "success": 0,
            "failed": 0,
            "skipped": 0
        }

        for msg in messages:
            msg_id = msg.get("id")

            if session_manager.is_email_processed(msg_id):
                result["skipped"] += 1
                continue

            session_manager.register_pending_email(msg_id)

            if self.process_email(msg, source=source):
                result["success"] += 1
            else:
                result["failed"] += 1

        return result

    def _is_spam(self, sender: str) -> bool:
        """Ki■m tra spam"""
        return any(pattern in sender for pattern in SPAM_PATTERNS)

    def _move_to_junk(self, message_id: str):
        """Di chuy■n email vào Junk"""
        move_url = f"https://graph.microsoft.com/v1.0/me/messages/{message_id}/move"
        move_body = {"destinationId": "junkemail"}
        try:
            requests.post(move_url, headers=self.headers, json=move_body, timeout=10)
        except Exception as e:
            print(f"[EmailProcessor] Move to junk error: {e}")

    def _save_attachments(self, message_id: str):
        """L■u file ■ính kèm"""
        url = f"https://graph.microsoft.com/v1.0/me/messages/{message_id}/attachments"
        try:
            resp = requests.get(url, headers=self.headers, timeout=10)
            if resp.status_code != 200:
                return

            attachments = resp.json().get("value", [])
            for att in attachments:
                if att.get("@odata.type") == "#microsoft.graph.fileAttachment":
                    file_name = att.get("name", "unknown_file")
                    content_bytes = att.get("contentBytes")

                    if content_bytes:
                        _, ext = os.path.splitext(file_name)
                        if not ext:
                            ext = ".bin"

                        att_path = os.path.join(ATTACH_DIR, f"{message_id}{ext}")
                        with open(att_path, "wb") as f:
                            f.write(base64.b64decode(content_bytes))

                        print(f"[EmailProcessor] Attachment saved: {att_path}")
        except Exception as e:
            print(f"[EmailProcessor] Save attachments error: {e}")

    def _forward_to_classifier(self, message: Dict):
        """G■i metadata ■■n MS2 Classifier"""
        try:
            metadata = {
                "id": message.get("id"),
                "subject": message.get("subject"),
                "bodyPreview": message.get("bodyPreview"),
                "hasAttachments": message.get("hasAttachments", False),
                "sender": message.get("from", {}).get("emailAddress", {}).get("address", ""),
                "receivedDateTime": message.get("receivedDateTime"),
                "raw_message": json.dumps(message)
            }

            response = requests.post(
                MS2_CLASSIFIER_BASE_URL,
                json=metadata,
                timeout=10
            )

            if response.status_code == 200:
                print(f"[EmailProcessor] Forwarded to MS2 successfully")
            else:
                print(f"[EmailProcessor] MS2 error: {response.status_code}")
        except requests.exceptions.RequestException as e:
            print(f"[EmailProcessor] Failed to connect to MS2: {e}")

    def _forward_to_persistence(self, message: Dict):
        """G■i metadata ■■n MS4 Persistence"""
        try:
            metadata = {
                "id": message.get("id"),
                "subject": message.get("subject"),
                "hasAttachments": message.get("hasAttachments", False),
                "sender": message.get("from", {}).get("emailAddress", {}).get("address", ""),
                "receivedDateTime": message.get("receivedDateTime"),
                "raw_message": json.dumps(message)
            }

            print(f"[EmailProcessor] Sending metadata to MS4: {metadata}")

            response = requests.post(
                f"{MS4_PERSISTENCE_BASE_URL}/metadata",
                json=metadata,
                timeout=10
            )

            if response.status_code in (200, 201):
                print(f"[EmailProcessor] Persisted to MS4 successfully")
            else:
                print(f"[EmailProcessor] MS4 error: {response.status_code}")
        except requests.exceptions.RequestException as e:
            print(f"[EmailProcessor] Failed to connect to MS4: {e}")
```