

CB????????????

? ??

?????? Cyberbiz API ?????????????????? Gmail API ?????????????????? LINE ??????????????????

? ?????

1. ??????
 - ?? Cyberbiz API (/v1/orders) ??????????open??
 - ??????????????????
2. ?????
 - ?? JSON ??????? /root/cb_order/orders/cb_order_YYYYMMDD.json ?
3. ?????
 - ???
 - ?????
 - ?????
 - ????????
 - ???????
 - ?????
4. **Email** ?????
 - ?? Gmail API?OAuth ??????????
 - ??????

□□□□□□ - YYYYMMDD

- ?????
 - ?????
 - ??????????????
 - ??????
 - ????????
 - ??????????
 - 2. **LINE** ??????????
 - ?? LINE Bot API ??????????????????
 - ??????????????????
-

? ??????????

- ??????
 - /root/cb_order
- ??????

- o /root/cb_order/orders/cb_order_YYYYMMDD.json
- **Gmail ??**
 - o credentials.json ?Gmail OAuth ??
 - o token.json ????????????
- **????**
 - o BASE_URL ?Cyberbiz API ??
 - o TOKEN ?Cyberbiz API Token
 - o TO_EMAILS ???????
 - o SENDER ????? Gmail ??

? ?????

```

□□□□□□□□20250331
□□□□□□□□18□□□□□:12□LINE Pay:5□□□□□□:1□
□□□□□□□□24500 □

□□□□□□□□□
- □□□□: 8 □
- □□□□: 6 □

□□□□□□□□
- □□□: 4 □

```

?? ?????

1. ?? TOKEN ? Gmail ?? (credentials.json)?
2. ??????

```
python3 cb_daily_report.py
```

3. ??????
 - ???????
 - ?? JSON ??
 - ?????????
 - ?? Email ??
 - ????????? LINE ??

? ??

- ??????????????????

- ?????????????/????
- ????????????

?? ??????

```
#!/usr/bin/env python3
import os
import json
import base64
import requests
from datetime import datetime, timedelta
from collections import defaultdict
from email.message import EmailMessage
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build

# === cron job json ===
os.chdir('/root/cb_order')

# ===  ===
BASE_URL = "https://app-store-api.cyberbiz.io"
TOKEN = ""
TO_EMAILS = [
    "wayne@lianhung.com.tw"
]
SENDER = "zfuntw@gmail.com"
SCOPES = ['https://www.googleapis.com/auth/gmail.send']

# ===  ===
yesterday = datetime.now() - timedelta(days=1)
start_time = yesterday.replace(hour=0, minute=0, second=0, microsecond=0)
end_time = yesterday.replace(hour=23, minute=59, second=59, microsecond=0)
start_time_str = start_time.strftime("%Y-%m-%d %H:%M:%S")
end_time_str = end_time.strftime("%Y-%m-%d %H:%M:%S")
report_date_str = yesterday.strftime("%Y%m%d")

# === orders/  ===
```

```

os.makedirs("orders", exist_ok=True)
json_path = f"orders/cb_order_{report_date_str}.json"

# === 测试数据 ===
params = {
    "start_time": start_time_str,
    "end_time": end_time_str,
    "statuses": "open"
}
HEADERS = {
    "Authorization": f"Bearer {TOKEN}",
    "Accept": "application/json"
}
response = requests.get(f"{BASE_URL}/v1/orders", headers=HEADERS, params=params)
if response.status_code != 200:
    print("❌ 测试数据")
    print(response.text)
    exit()
order_list = response.json()

# === 测试数据 ===
all_order_details = []
for order in order_list:
    order_id = order.get("id")
    if not order_id:
        continue
    detail_res = requests.get(f"{BASE_URL}/v1/orders/{order_id}", headers=HEADERS)
    if detail_res.status_code == 200:
        all_order_details.append(detail_res.json())
        print(f"✅ 测试数据 {order_id}")
    else:
        print(f"⚠️ 测试数据 {order_id}")

# === 生成 JSON 文件 ===
with open(json_path, "w", encoding="utf-8") as f:
    json.dump(all_order_details, f, ensure_ascii=False, indent=2)

# === 统计订单数量和金额 ===
total_orders = len(all_order_details)
total_amount = 0

```

```

product_summary = defaultdict(int)
gift_summary = defaultdict(int)
payment_methods = defaultdict(int)

for order in all_order_details:
    payment = order.get("payment_method", " ")
    payment_methods[payment] += 1

    for item in order.get("line_items", []):
        title = item.get("title", " ")
        qty = item.get("quantity", 0)
        price = item.get("total_price_after_discounts", item.get("price", 0))
        if price == 0:
            gift_summary[title] += qty
        else:
            product_summary[title] += qty
            total_amount += price

# ===  =====  ===
payment_str = "".join([f"{k}:{v}" for k, v in sorted(payment_methods.items(), key=lambda x: -x[1])])

# ===  =====  ===
report_lines = [
    f"===== {report_date_str}",
    f"===== {total_orders} {payment_str} ",
    f"===== {total_amount:.0f} ",
    "",
    "===== "
]

for title, qty in sorted(product_summary.items(), key=lambda x: -x[1]):
    report_lines.append(f"- {title}: {qty} ")

if gift_summary:
    report_lines.append("")
    report_lines.append("===== ")
    for title, qty in sorted(gift_summary.items(), key=lambda x: -x[1]):
        report_lines.append(f"- {title}: {qty} ")

email_body = "\n".join(report_lines)

```

```

# === Gmail API ===
creds = None
if os.path.exists('token.json'):
    creds = Credentials.from_authorized_user_file('token.json', SCOPES)
if not creds or not creds.valid:
    if creds and creds.expired and creds.refresh_token:
        creds.refresh(Request())
    else:
        flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
        creds = flow.run_local_server(port=0)
with open('token.json', 'w') as token:
    token.write(creds.to_json())

service = build('gmail', 'v1', credentials=creds)

# === Email ===
message = EmailMessage()
message.set_content(email_body)
message['To'] = ", ".join(TO_EMAILS)
message['From'] = SENDER
message['Subject'] = f"{} - {report_date_str}"

encoded_message = base64.urlsafe_b64encode(message.as_bytes()).decode()
send_message = service.users().messages().send(userId="me", body={"raw":
encoded_message}).execute()

print(f"\n ID: {send_message['id']}")

# === LINE ===
#from linebot import LineBotApi
#from linebot.models import TextSendMessage
#from dotenv import load_dotenv

#
#load_dotenv()

#print("DEBUG - LINE_CHANNEL_ACCESS_TOKEN:", os.getenv("LINE_CHANNEL_ACCESS_TOKEN"))
#print("DEBUG - LINE_GROUP_IDS:", os.getenv("LINE_GROUP_IDS"))

```

```

#LINE_TOKEN = os.getenv("LINE_CHANNEL_ACCESS_TOKEN")
#LINE_GROUP_IDS = os.getenv("LINE_GROUP_IDS", "").split(",")
#LINE_GROUP_IDS = [gid.strip() for gid in LINE_GROUP_IDS if gid.strip()]

#if LINE_TOKEN and LINE_GROUP_IDS:
#    try:
#        line_bot_api = LineBotApi(LINE_TOKEN)

#        # LINE 500
#        short_report = "\n".join(report_lines)
#        max_len = 480
#        if len(short_report) > max_len:
#            short_report = short_report[:max_len] + "\n... "
#
#        for gid in LINE_GROUP_IDS:
#            try:
#                line_bot_api.push_message(
#                    gid,
#                    TextSendMessage(text=short_report)
#                )
#                print(f" LINE {gid}")
#            except Exception as e:
#                print(f" LINE {gid} {e}")
#
#    except Exception as e:
#        print(f" LINE {e}")
#else:
#    print(" LINE_CHANNEL_ACCESS_TOKEN LINE_GROUP_IDS")

```

Revision #2

Created 1 August 2025 04:58:28 by Wayne

Updated 1 August 2025 05:26:55 by Wayne