

2FA (2FA)

2FA Overview

OmniCRM 2FA (2FA) is a security feature that requires users to provide two forms of authentication (something they know and something they have) to access the system. This is implemented using Time-based One-time Passwords (TOTP).

2FA is configured in the RBAC <rbac> 2FA section of the Authentication Flows <authentication_flows> configuration.

2FA

2FA Settings

- 2FA is enabled for all users.
- 2FA is required for all users.
- 2FA is required for all users.
- 2FA is required for all users. Supported authenticators include Google Authenticator, Authy, and Microsoft Authenticator. TOTP is also supported.

2FA Configuration

2FA Configuration

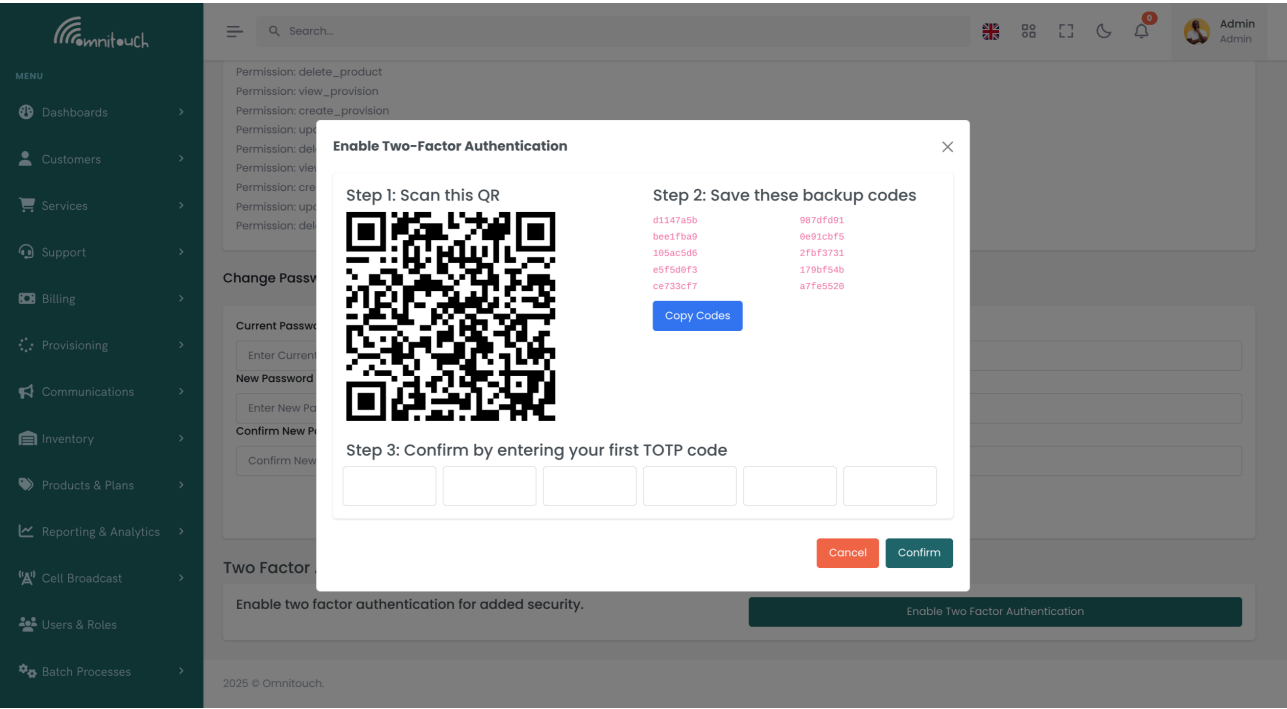
- 2FA is enabled for all users.
- 2FA is required for all users.
- 2FA is required for all users.
- 2FA is required for all users.

2FA

2FA

2FA

1. 2FA
2. 2FA
3. 2FA
4. 2FA
5. 2FA



2FA

2FA

2FA

- 2FA
- 2FA

- □□□□□□□□□□□□□□

--	--	--	--	--

```
totp_secret
2FA2FA
```

API ☐ ☐

2FA

POST /2fa/enable/user/{user_id}

```
{
  "password": "current_password"
}
```

URI

2FA

POST /2fa/verify-setup/user/{user_id}

```
{
  "code": "123456"
}
```

2FA

POST /2fa/verify/user/{user id}

```
{
  "code": "123456"
}
```

API Endpoints

API Endpoints

POST /2fa/backup-codes/regenerate/user/{user_id}

API Endpoints

API Endpoints

- API Endpoints for 2FA
- API Endpoints for 2FA
- API Endpoints for 2FA

API Endpoints

API Endpoints for TOTP: Google Authenticator, Authy, Microsoft Authenticator, 1Password

API Endpoints for 2FA

API 概念

API 是应用程序编程接口，是 **Web** 应用程序与 OmniCRM API 之间的桥梁。

API

API 的概念是 `concepts_api`

API

API 是应用程序编程接口，是应用程序与 OmniCRM API 之间的桥梁。

- 应用程序编程接口
- 应用程序
- 应用程序
- 应用程序 cron 任务
- 应用程序

JWT 令牌是应用程序/应用程序 API 令牌，用于身份验证。

API 概念

API

API

API `MANAGE_API_KEYS` 令牌

API 概念

API 是应用程序编程接口，是应用程序与 OmniCRM API 之间的桥梁。

API

- API 是应用程序编程接口，是应用程序与 OmniCRM API 之间的桥梁。

- 文档 - 文档模板文档
- **API** 文档 - 文档模板文档
- 文档 - 文档模板文档
- 文档 - 文档模板文档
- 文档 - 文档模板文档
- 文档 - 文档模板文档

□ □ □ □ □

□ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □

- **API** - API ــ
- ــــــــــــــــ - ــــــــــــــــــــــــ
- ــــــــــــــــ - ــــــــــــــــ 30 ــــــــــــــــــــــــ
- ــــــــــــــــ - ــــــــــــــــــــــــــــــــ

API

1000 “API 00”

API +

2

[illegible]

□ □ □

□□□□“□□□□”□

□□□ _____

□□□ - □□□□□□□□

□□□□□[□□□□□]

□□□ - □□□□□□□□

□□□□ □□□□ □ □□□□ □ □□□□ □ □□ □ □□□□ ☒ □□□□□□ □□□

[□□] [□□□□□]

□□□□□

□□□□□□

- □□□□□□□□□□
- □□□“□□□”□“□□□□”□“□□”
- □□□□□□□□□□□□

□□□□□□

- □□□□□□
- □□□“□ Ansible □□□□□□□”□“□□□□□□□”
- □□□□□□□□□□□□□□□

□□□□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□90 □□ 1 □□

□□

- □□□□□□□□□□“□□□”□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□
- □□□□□□ □□□□□□□□□

□□ **3**□□□□□□□□□

□□ “□□□□□” □□□□□□□□□□□□□□ API □□□

 □□□□□□□□ - □□□□□□□□□

sk_live_a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0

[□□□□□□□□]

[]

API

sk_live_...XYZ

4

API

```
curl -X GET "https://yourcrm.com/crm/customers" \
-H "X-API-KEY: sk_live_a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0"
```

```
export
CRM_API_KEY="sk_live_a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0"
```

API

-
-
-
-
-
-

如何调用 API

调用 API 的步骤如下：

1. 获取 API 地址
2. 构造请求参数
3. 发送请求

例如：

获取 API 地址

构造请求参数

发送请求

- 获取 API 地址
- 构造请求参数
- 发送请求
- 接收响应

如何调用 API

调用 API 的步骤如下：

1. 获取 API 地址
2. 构造请求参数
3. 发送请求

例如：

- 获取 API 地址
- 构造请求参数 401 Unauthorized
- 发送请求
- 接收响应 - 401 Unauthorized

例如：

- 获取 API 地址

- 環境問題
- 経済問題
- 社会問題

API 環境問題

API 環境問題

1. 環境問題 : 環境問題
2. “環境”
3. 環境問題
4. “環境”

環境問題

- 環境
- 環境
- 環境
- 環境

環境問題

- 環境問題
- 環境
- 環境

API 環境問題

API 環境問題

環境

- 環境問題
- 環境問題
- 環境
- 環境

□□□□

- □□□□□□□□ 30 □□□□
- □□□□□/□□□□
- □□□□□□□□□□

□□

- □□□□□□
- □□□□□□
- □□□□□□□
- □□□□□□□□□□

□□□

- □□□□□/□□
- □□□□□
- □□□□□□□□□□

□□□□□□

API □□□□□□□

□□□

□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□

- □□□□□
- □□□□
- □□□□□□□□□□ 30 □□
- □□□□

□□□

API 管理

- API
- API
- API
- API

API 管理

API 管理

- API 管理 32 管理
- API 管理
- API 管理 `sk_live_` 管理

API 管理

CRM 管理

- API 管理
- API 管理
- API 管理
- API 管理

API 管理

- API 管理
- API 管理 AWS Secrets Manager HashiCorp Vault
- API 管理
- API 管理 90-365 管理

API 管理

- API 管理 - API 管理
- API 管理

- 認證/授權
- 認證

認證

- 認證 API 認證
- 認證
- “認證”
- 認證

認證

認證

1. 認證
2. 認證
3. 認證
4. 認證

認證

API “401 Unauthorized”

- 認證
- 認證
 - 認證
 - 認證
 - 認證
 - 認證 X-API-KEY 認證 Authorization

“API ”

- 認證
- 認證
 - API 認證

- 時間範囲を指定する
- 時間範囲を指定する

API 呼び出し

- 時間範囲を指定する30 分間
- 時間範囲を指定する
 - 時間範囲を指定する
 - 時間範囲を指定する
 - 時間範囲を指定する

API 呼び出し

- 時間範囲を指定する
- 時間範囲を指定する
 - 時間範囲を指定する
 - 時間範囲を指定する/時間
 - 時間範囲を指定する???

API 呼び出し

API 呼び出し API 呼び出し

API 呼び出し

```
GET /crm/api-keys
Authorization: Bearer <admin-token>
```

API 呼び出し

POST /crm/api-keys

Authorization: Bearer <admin-token>

Content-Type: application/json

```
{
  "name": "New Integration",
  "description": "Third-party billing sync",
  "expiry_date": "2026-01-10",
  "permissions": ["view_customer", "view_service"]
}
```

📄

```
{
  "api_key_id": 123,
  "name": "New Integration",
  "api_key": "sk_live_a1b2c3d4e5f6g7h8i9j0",
  "status": "active",
  "created": "2025-01-10T10:00:00Z",
  "expiry_date": "2026-01-10T23:59:59Z"
}
```

📄 **API** 📄

DELETE /crm/api-keys/{api_key_id}

Authorization: Bearer <admin-token>

📄 **API** 📄

PATCH /crm/api-keys/{api_key_id}

Authorization: Bearer <admin-token>

Content-Type: application/json

```
{
  "name": "Updated Name",
  "expiry_date": "2026-12-31"
}
```

📄📄📄📄

📄 **1**📄📄📄📄📄

📄📄 Ansible 📄📄📄📄📄📄 API 📄📄

1. 📄📄 API 📄📄 → 📄📄
2. 📄📄“Ansible 📄📄📄📄”
3. 📄📄“📄📄📄📄📄📄”
4. 📄📄📄📄📄📄/📄📄📄📄📄📄/📄📄📄
5. 📄📄1 📄
6. 📄📄📄📄📄📄 Ansible `crm_config.yaml`

📄 **2**📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄

1. 📄📄“📄📄📄 - QuickBooks”
2. 📄📄📄📄📄📄📄📄📄📄📄
3. 📄📄90 📄📄📄📄📄📄
4. 📄📄📄📄📄📄📄

📄 **3**📄📄📄📄

📄 Prometheus/Grafana 📄📄📄📄📄📄📄

1. 📄📄“📄📄 - Grafana”
2. 📄📄📄📄📄📄📄📄
3. 📄📄📄📄📄📄📄📄📄📄
4. 📄 Grafana 📄📄📄📄

📄 **4**📄📄📄 **API**

📄📄📄📄📄📄📄📄📄📄

1. 📄📄“📄📄📄📄📄”
2. 📄📄📄📄📄📄📄📄📄📄📄📄📄📄
3. 📄📄180 📄

4. 000000 API 000000

000000

- `concepts_api` - API 00000000
- `rbac` - 00000000000000
- `2fa` - 00000000000000

API

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

API

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

1. API

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

- external_crm_id = "SF-12345" Salesforce ID
- legacy_system_id = "OLD-CRM-789" Legacy System ID
- hubspot_contact_id = "12345678" HubSpot ID

2. API

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

- billing_method = "quarterly" Billing Method
- auto_provision = "true" Auto Provision
- support_tier = "premium" Support Tier
- credit_limit = "10000" Credit Limit

3. API

Customer Tags <administration_tags> Customers, Contacts, Sites & Services <basics_customers>

- gdpr_consent_date = "2025-01-01" GDPR Consent Date
- tax_exempt = "true" Tax Exempt

- `regulatory_entity` = "FCC-123456" 00000000

4. 000000

00000000

- `preferred_contact_method` = "email" 000000
- `account_manager` = "<john.smith@company.com>" 0000000000
- `onboarding_date` = "2025-01-15" 0000000000
- `churn_risk_score` = "0.23" 000000

5. 0000

000000000000

- `radius_username_format` = "email" 0000RADIUS000
- `vlan_id` = "100" 000000
- `ipv6_enabled` = "true" 000000

0000000000

00000000

- 0000000000000000
- 000000
- 000000000000
- 0000000000000000
- 00000000000000

0000000000

- 000000000000000000000000
- 000000000000
- 000000000000
- 0000 0000000000

UI 仕様書

概要

目的

- システム概要
- 利用シーン
- 機能要件
 - 機能A
 - 機能B
 - 機能C
 - 機能D

詳細仕様

機能A

- 機能A概要
- 利用シーン
- 機能A詳細
- 入力/出力
 - 外部システム連携/外部ID external_crm_id
 - 外部システム連携/外部ID SF-12345
- その他

機能B

- 機能B概要 external_system_id ✓
- 機能B詳細 external system id ✗
- 機能B詳細
- 機能B詳細

その他

備考

1. 資料の取得
2. 検索条件の指定
3. 結果の表示
4. 結果の加工
5. 結果の出力
6. 結果の保存

例

データベースからデータを取得し、加工して出力する。❖❖

例

例

1. 資料の取得
2. 検索条件の指定
3. 結果の表示
4. 結果の加工
5. 結果の出力

例

データベースからデータを取得し、加工して出力する。

例

API

データベースAPIの使い方

例

例 PUT /crm/attribute/

POST create_customer_attribute

Request

```
{
  "customer_id": 123,
  "attribute_name": "external_crm_id",
  "attribute_value": "SF-12345"
}
```

Response

```
{
  "attribute_id": 456,
  "customer_id": 123,
  "attribute_name": "external_crm_id",
  "attribute_value": "SF-12345",
  "created": "2025-01-04 10:30:00",
  "last_modified": "2025-01-04 10:30:00"
}
```

PATCH

PATCH /crm/attribute/attribute_id/{attribute_id}

POST update_customer_attribute

Request

```
{
  "attribute_value": "SF-54321"
}
```

GET ID

GET /crm/attribute/attribute_id/{attribute_id}

POST view_customer_attribute

####

```
{
  "attribute_id": 456,
  "customer_id": 123,
  "attribute_name": "external_crm_id",
  "attribute_value": "SF-12345",
  "created": "2025-01-04 10:30:00",
  "last_modified": "2025-01-04 10:30:00"
}
```

####ID#####

GET /crm/attribute/customer_id/{customer_id}

view_customer_attribute

####

```
[
  {
    "attribute_id": 456,
    "customer_id": 123,
    "attribute_name": "external_crm_id",
    "attribute_value": "SF-12345",
    "created": "2025-01-04 10:30:00",
    "last_modified": "2025-01-04 10:30:00"
  },
  {
    "attribute_id": 457,
    "customer_id": 123,
    "attribute_name": "billing_method",
    "attribute_value": "quarterly",
    "created": "2025-01-04 10:35:00",
    "last_modified": "2025-01-04 10:35:00"
  }
]
```

DELETE

DELETE /crm/attribute/attribute_id/{attribute_id}

delete_customer_attribute

```
{  
  "result": "success"  
}
```

DELETE

DELETE /crm/attribute/attribute_id/{attribute_id}

```
import requests  
  
customer_id = 123  
attributes = [  
    {"attribute_name": "external_crm_id", "attribute_value": "SF-12345"},  
    {"attribute_name": "billing_method", "attribute_value": "quarterly"},  
    {"attribute_name": "support_tier", "attribute_value": "premium"}  
]  
  
for attr in attributes:  
    attr["customer_id"] = customer_id  
    requests.put(  
        "https://api.example.com/crm/attribute/",  
        json=attr,  
        headers={"Authorization": "Bearer YOUR_TOKEN"}  
    )
```

API 実装

API 実装のサンプルコード

```
# API 実装のサンプルコード
customers = requests.get("https://api.example.com/crm/customer/").json()

for customer in customers:
    attributes = requests.get(
        f"https://api.example.com/crm/attribute/customer_id/{customer['customer_id']}"
    ).json()

    # 属性のチェック
    for attr in attributes:
        if attr['attribute_name'] == 'support_tier' and
        attr['attribute_value'] == 'premium':
            print(f"Premium customer: {customer['customer_name']}")
```

API

API の仕様を確認するためのテストコード

API の仕様

1. 入力パラメータ

- 外部システム ID (external_system_id) ✓
- 請求方法 (billing_method) ✓ / 方法 (method) ✗
- 顧客 ID
- 顧客名

2. 出力パラメータ

- 顧客 ID (150 文字以内)
- 顧客名 ("true"/"false" 形式)
- ISO 8601 形式のタイムスタンプ "2025-01-04 10:30:00"
- JSON 形式の出力

3. 目的

- 目的を達成するための具体的な目標を設定する
- 目標達成のための具体的な計画を立てる
- 計画の実行状況を定期的に確認する

4. 方法

- 目的達成のための具体的な方法を考える
- 計画の実行状況を定期的に確認する
- 計画の実行状況を定期的に確認する

5. 結果

- 目的達成のための具体的な方法を考える
- 計画の実行状況を定期的に確認する
- 計画の実行状況を定期的に確認する `api_key_v2`

目的達成のための具体的な方法を考える

計画の実行状況を定期的に確認する

計画の実行状況を定期的に確認する

```
# 更新旧属性
PUT /crm/attribute/
{
  "customer_id": 123,
  "attribute_name": "legacy_crm_id",
  "attribute_value": "OLD-12345"
}

# 更新新属性
PUT /crm/attribute/
{
  "customer_id": 123,
  "attribute_name": "migrated_date",
  "attribute_value": "2025-01-04"
}
```

更新旧属性

更新新属性

```
# 更新账单周期
PUT /crm/attribute/
{
  "customer_id": 123,
  "attribute_name": "billing_cycle",
  "attribute_value": "quarterly"
}

# 验证更新后的属性
attributes = GET /crm/attribute/customer_id/123
billing_cycle = next(
    (a['attribute_value'] for a in attributes if
     a['attribute_name'] == 'billing_cycle'),
    'monthly' # 默认值
)
```

更新账单周期

验证更新后的属性


```
# IPv6
PUT /crm/attribute/
{
  "customer_id": 123,
  "attribute_name": "feature_ipv6_enabled",
  "attribute_value": "true"
}
```

API

API endpoints

- `view_customer_attribute` - GET
- `create_customer_attribute` - POST
- `update_customer_attribute` - PUT
- `delete_customer_attribute` - DELETE

API uses `rbac` for permissions

UI

UI components

- `API` endpoints
- `view_customer_attribute` component
- `view_customer_attribute` component

UI components

- `update_customer_attribute` component
- `attribute_id` component
- `attribute_value` component

UI components

- `attribute_value` component

-
- 

□□□□□□

- 150
-
-

OmniCRM

OmniCRM 是一個基於微服務架構的 CRM 系統，旨在為企業提供全面的客戶關係管理解決方案。

安裝

OmniCRM 安裝步驟如下：

1. 克隆代碼

- 克隆 OmniCRM-API/crm_config.yaml
- 克隆 YAML
- 克隆 資料庫 API
- 克隆 資料庫

2. 配置

- 配置 OmniCRM-UI/.env
- 配置 資料庫
- 配置 資料庫 UI
- 配置 資料庫

配置 (crm_config.yaml)

crm_config.yaml 是一個用於配置 OmniCRM 的 YAML 文件。

文件路徑：`/OmniCRM/OmniCRM-API/crm_config.yaml`

Database

```
database:
  username: omnitouch
  password: omnitouch2024
  server: localhost
```

Notes

- `username` - MySQL username
- `password` - MySQL password
- `server` - IP address of MySQL server (default: localhost)

Environment

- Database name: `omnicrm`
- Port: 3306 (MySQL default port)
- Connection string: `mysql+pymysql://username:password@server/omnicrm`

Make sure the database is running and accessible before proceeding.

Set the `DB_HOST` environment variable to the IP address of the MySQL server (e.g., `localhost` or `db` in Docker).

Services

```
service_types:
  - omnicharge
  - mobile
  - fixed
  - fixed-voice
  - hotspot
  - dongle
```

Make sure the services are running and accessible before proceeding.

Deployment

- `docker-compose up`

- 設定する必要がある項目
- 設定方法
- 確認方法

設定する必要がある項目

HSS 設定

HSS 設定

```
hss:  
  hss_peers:  
    - 'http://10.179.2.140:8080'  
  apn_list: "1,2,3,4,5,6"
```

設定

- `hss_peers` - HSS の IP アドレス
- `apn_list` - APN ID のリスト

設定を確認する

Mailjet 設定

OmniCRM から Mailjet 設定を確認する

```
mailjet:
  api_key: your_mailjet_api_key
  api_secret: your_mailjet_api_secret

# 顧客からのメール
api_crmCommunicationCustomerWelcome:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977509
  subject: "Welcome to YourCompany"

# 顧客からのメール
api_crmCommunicationCustomerInvoice:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 6759851
  subject: "Your Invoice - "

# 顧客からのメール
api_crmCommunicationCustomerInvoiceReminder:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 5977570
  subject: "Invoice Payment Reminder"

# 顧客からのメール/顧客からのメール
api_crmCommunicationUserWelcome:
  from_email: "admin@yourcompany.com"
  from_name: "Your Company Admin"
  template_id: 6118112
  subject: "Welcome to the Team"

# 顧客からのメール
api_crmCommunicationUserPasswordReset:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 6735666
  subject: "Password Reset Request"

# 顧客からのメール
api_crmCommunicationUserPasswordResetSuccess:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
```

```
template_id: 6118378
subject: "Password Reset Successful"
```

密碼重置

```
api_crmCommunicationUserPasswordChange:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 6118423
  subject: "Password Changed"
```

驗證郵件

```
api_crmCommunicationEmailVerification:
  from_email: "verify@yourcompany.com"
  from_name: "Your Company Verification"
  template_id: 6267350
  subject: "Verify Your Email Address"
```

服務平衡已到期

```
api_crmCommunicationsBalanceExpired:
  from_email: "alerts@yourcompany.com"
  from_name: "Your Company Alerts"
  template_id: 7238252
  subject: "Service Balance Expired"
```

服務平衡低

```
api_crmCommunicationsBalanceLow:
  from_email: "alerts@yourcompany.com"
  from_name: "Your Company Alerts"
  template_id: 7238263
  subject: "Low Balance Warning"
```

Mailjet 設定

1. 前往 Mailjet 控制台 (<<https://app.mailjet.com>>)
2. 點選 **Transactional** → **Templates**
3. 點選 **Get API Key**
4. 點選 **Template ID**
5. 點選 OmniCRM 設定頁面
6. 在 `crm_config.yaml` 檔案填入 ID

完成設定

変数テンプレート

- `{{customer_name}}` - 顧客名
- `{{service_name}}` - サービス名
- `{{invoice_id}}` - 請求ID
- `{{invoice_amount}}` - 請求金額
- `{{due_date}}` - 支払期限
- `{{reset_link}}` - リセット URL
- `{{verification_link}}` - 検証 URL
- `{{balance}}` - 残高
- `{{expiry_date}}` - 有効期限

例

```
provisioning:  
  failure_list: ['admin@yourcompany.com', 'ops@yourcompany.com']
```

説明

- `failure_list` - Ansible 実行失敗時の連絡先
- 必須項目
- 複数指定可能

例

```
invoice:  
  template_filename: 'yourcompany_invoice_template.html'
```

説明

請求書は PDF 形式で Jinja2 HTML テンプレート

を `/OmniCRM-API/invoice_templates/`

から取得し、PDF 形式で出力

CRM 設定 URL

```
crm:
  base_url: 'http://localhost:5000'
```

設定

- Ansible 設定ファイル API 設定
- 設定ファイルに CRM 設定
- 設定ファイル API 設定 URL 設定

設定

- 設定 http://localhost:5000
- 設定 https://api.yourcompany.com
- Docker http://omnicrm-api:5000 設定

設定

- 設定
- 設定 URL
- 設定

OCS 設定 CGRates 設定

```
ocs:
  ocsApi: 'http://10.179.2.142:8080/api'
  ocsTenant: 'mnc380.mcc313.3gppnetwork.org'
  cgrates: 'localhost:2080'
```

設定

- ocsApi - OCS API 設定
- ocsTenant - OCS 設定
- cgrates - CGRates JSON-RPC API 設定

設定 設定

SMSC 設定

```
smc:  
  source_msidn: 'YourCompany'  
  smc_url: 'http://10.179.2.216/SMSc/'  
  api_key: 'your_smc_api_key'
```

設定

- SMS 認証 2FA 設定
- `source_msidn` - 認証 ID
- `smc_url` - SMSC API URL
- `api_key` - SMSC API キー

設定

```
cbc_url: 'http://10.179.1.113:8080'
```

設定 CBC API

設定 `features_cell_broadcast`

JWT 設定

```
jwt_secret: 'CHANGE_ME_ON_DEPLOYMENT' # Ansible 設定
```

設定

- JWT 設定
- 認証 - 設定
- 認証/認証
- 認証

設定 **Ansible** 設定

📄 Ansible 📄JWT 📄📄📄📄📄

1. 📄📄📄📄 Ansible 📄📄📄📄 64 📄📄📄📄📄📄256 📄📄
2. 📄📄📄 Ansible 📄📄📄📄📄📄📄
3. 📄📄📄📄📄📄📄📄 - 📄📄📄📄📄📄📄📄

📄📄📄

- Git 📄📄📄📄📄📄📄
- 📄📄📄📄📄📄📄
- 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄
- 📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄 **Ansible** 📄📄

📄📄 Ansible 📄📄📄📄📄📄📄📄

```
# 📄📄📄📄📄📄📄  
python3 -c "import secrets; print(secrets.token_hex(32))"  
# 📄  
openssl rand -hex 32
```

📄📄 `crm_config.yaml` 📄

```
jwt_secret: 'your_generated_secret_here'
```

📄📄

- 📄📄📄📄📄📄📄 `CHANGE_ME_ON_DEPLOYMENT`
- 📄📄📄📄 - 📄📄📄📄📄📄📄📄📄📄
- 📄📄📄📄📄📄📄📄📄📄📄📄📄
- 📄📄📄📄📄📄📄📄📄📄📄📄

Stripe 設定

```
stripe:
  secret_key: 'sk_live_XXXXXXXXXX'
  publishable_key: 'pk_live_XXXXXXXXXX'
  currency: 'aud'
  statement_descriptor_suffix: 'YOURCOMPANY'
```

設定

- `secret_key` - Stripe の API 秘密鍵
- `publishable_key` - Stripe の公開鍵
- `currency` - ISO 4217 コード (aud, usd, gbp, eur など)
- `statement_descriptor_suffix` - 明細書の接尾辞

テスト

- テスト用 `sk_test_...` と `pk_test_...` を使用する
- 本番用 `sk_live_...` と `pk_live_...` を使用する

ファイル

- 明細書の接尾辞を "YOURCOMPANY"
- 22 桁の番号
- 明細書の形式
- PDF 明細書のファイル名: `YOURCOMPANY_12345.pdf`

Payment Vendor Integrations
<integrations_payment_vendors>

API の IP 設定

API の IP 設定

```
api_keys:
  "YOUR_API_KEY_1":
    roles: ["admin"]
    ips: ["127.0.0.1", "::1"]
  "YOUR_API_KEY_2":
    roles: ["customer_service_agent_1"]
    ips: ["127.0.0.1", "::1", "10.0.1.0/24"]

# IP 白名单 API 接口
ip_whitelist:
  "10.179.2.142":
    roles: ["admin"]
```

配置

- 配置 API 接口
- 配置 IP 白名单
- 配置 API 接口
- 配置 API 接口

配置 API 接口

```
openssl rand -base64 48
```

配置

- `admin` - 配置 API 接口
- 配置 RBAC 配置 API 接口

配置 API 接口

- 配置 API 接口 32 位
- 配置 IP 白名单
- 配置 API 接口
- 配置 API 接口
- 配置 API 接口

配置 API 接口

- 設定可能な IP 範囲
- 設定可能な IP 127.0.0.1:::1
- 設定可能な API 範囲
- 設定可能な API 範囲

設定可能な API 範囲 `administration_api_keys` と `concepts_api`

環境変数 (.env)

React UI の `OmniCRM-UI/.env` を設定する

パス `/OmniCRM/OmniCRM-UI/.env`

API 設定

```
# Google Maps API 設定
VITE_GOOGLE_API_KEY=your_google_api_key

# Stripe 設定
VITE_STRIPE_PUBLISHABLE_KEY=pk_test_xxxxx

# npm start 時の設定
BROWSER=none
```

設定

`VITE_STRIPE_PUBLISHABLE_KEY` を `crm_config.yaml` の `stripe.publishable_key` に設定

環境変数

```
# 会社名
VITE_COMPANY_NAME="ShellFone"
VITE_PORTAL_NAME="ShellManager"
VITE_SELF_CARE_NAME="ShellCare"
VITE_COMPANY_TAGLINE="Phones with Shells"
```

環境変数 UI 環境変数

- `COMPANY_NAME` - 会社名
- `PORTAL_NAME` - ポータル/管理画面名“ShellManager”
- `SELF_CARE_NAME` - セルフケア画面名“ShellCare”
- `COMPANY_TAGLINE` - 会社タグライン

環境変数

```
# 言語
# 言語コード: ar, ch, en, fr, gr, it, ru, sp
VITE_DEFAULT_LANGUAGE=en
VITE_LOCALE="en-GB"

# 地域
VITE_DEFAULT_LOCATION="Sydney, Australia"
VITE_DEFAULT_COUNTRY="Australia"

# 通貨
VITE_CURRENCY_CODE="GBP"
VITE_CURRENCY_SYMBOL="£"
```

言語コード

- `ar` - アラビア語
- `ch` - 中国語
- `en` - 英語
- `fr` - フランス語
- `gr` - ギリシ語

- `it` - イタリア
- `ru` - ロシア
- `sp` - スペイン

通貨コード

- USD - \$
- GBP - £
- EUR - €
- AUD - \$
- CAD - \$

例: `crm_config.yaml` の `stripe.currency`

色コード

```
# 色コード
VITE_PRIMARY_COLOR=#405189

# 色コード
# VITE_SECONDARY_COLOR=#2bFFcf
# VITE_TERTIARY_COLOR=#1a9fbf
# VITE_SUCCESS_COLOR=#28a745
# VITE_INFO_COLOR=#17a2b8
# VITE_WARNING_COLOR=#ffc107
# VITE_DANGER_COLOR=#dc3545
```

色コード

- `VITE_PRIMARY_COLOR` - 色コード
- `VITE_SECONDARY_COLOR` - 色コード
- `VITE_TERTIARY_COLOR` - 色コード
- `VITE_SUCCESS_COLOR` - 色コード #28a745
- `VITE_INFO_COLOR` - 色コード #17a2b8
- `VITE_WARNING_COLOR` - 色コード #ffc107
- `VITE_DANGER_COLOR` - 色コード #dc3545
- `VITE_LIGHT_COLOR` - 色コード #f8f9fa

- `VITE_DARK_COLOR` - `hex:#343a40`
- `VITE_PRIMARY_DARK_COLOR` - `hex:#000000`

hex `#RRGGBB`

font-family

- font-family
- font-family
- font-family
- font-family
- font-family

font-family

```
# font-family
# font-family Inter, Roboto, Open Sans, Lato, Quicksand, Poppins, Nunito,
Montserrat, Work Sans, Source Sans Pro, Raleway, Ubuntu, Josefin Sans,
HKGrotesk
# font-family Merriweather, Lora, Playfair Display
# font-family System, sans-serif
# font-family Quicksand
VITE_FONT_FAMILY=Quicksand
```

font-family **OmniCRM-UI** font-family

- font-family - font-family
- font-family - font-family
- font-family - font-family
- font-family - Google Fonts, Adobe Fonts, CDN, font-family
- font-family - font-family
- font-family - font-family

font-family

font-family

- Inter Roboto Open Sans Lato Quicksand Poppins Nunito Montserrat Work Sans Source Sans Pro Raleway Ubuntu Josefin Sans HKGrotesk

Font

- Merriweather Lora Playfair Display

Font

- System - Font

Font

Font

Font

1. Font OmniCRM-UI/src/assets/fonts/your-font-name/

- Font WOFF2 Font
- Font 300 400 500 600 700 Font
- Font your-font-name-300.woff2 your-font-name-400.woff2

2. Font OmniCRM-UI/src/assets/scss/fonts/_google-fonts.scss Font @font-face

```
//
// 自定义字体 - 400
//

@font-face {
  font-family: 'Your Font Name';
  font-style: normal;
  font-weight: 400;
  font-display: swap;
  src: url("../fonts/your-font-name/your-font-name-400.woff2") format('woff2');
}

@font-face {
  font-family: 'Your Font Name';
  font-style: normal;
  font-weight: 700;
  font-display: swap;
  src: url("../fonts/your-font-name/your-font-name-700.woff2") format('woff2');
}
```

3. 在 .env 文件中配置

```
VITE_FONT_FAMILY=Your Font Name
```

在 main.ts 中

- 300 - 超细
- 400 - 细
- 500 - 中
- 600 - 粗
- 700 - 超粗

Web 字体

目前 6 种 Web 字体格式

```
# Web 앱 1 GitHub
VITE_WEB_APP_1_NAME="GitHub"
VITE_WEB_APP_1_URL="https://github.com"
VITE_WEB_APP_1_ICON_PATH="resources/webapp_icons/github.png"

# Web 앱 2 Xero
VITE_WEB_APP_2_NAME="Xero"
VITE_WEB_APP_2_URL="https://go.xero.com/"
VITE_WEB_APP_2_ICON_PATH="resources/webapp_icons/xero.png"

# Web 앱 3-6
# 앱 이름 URL 아이콘 경로
```

예시

- `VITE_WEB_APP_N_NAME` - 앱 이름
- `VITE_WEB_APP_N_URL` - 앱 URL
- `VITE_WEB_APP_N_ICON_PATH` - 앱 아이콘 경로 public/

예시: GitHub, Xero, Monday.com, Gmail, MailJet, Slack

Grafana 설정

OmniCRM에 Grafana를 연결하려면 `grafana_access` 변수에 `/grafana`를 추가하고 Grafana의 기본 URL을 OmniCRM에 설정합니다.

```
# Grafana 설정
# 기본 ID - /grafana/d/ID/name URL
VITE_GRAFANA_DASHBOARD_IDS=abc123,def456,ghi789

# 기본 ID 라벨
VITE_GRAFANA_DASHBOARD_LABELS=System Metrics,User Analytics,Network Performance
```

예시

- `VITE_GRAFANA_DASHBOARD_IDS` - URL에 Grafana 기본 ID를 추가합니다.
- `VITE_GRAFANA_DASHBOARD_LABELS` - 기본 ID에 라벨을 추가합니다.

ダッシュボード ID

Grafana ダッシュボード URL を `/grafana/d/abc123/dashboard-name` ID を `/d/` の後に追加します `abc123`

例

ダッシュボード ID を OmniCRM の “ダッシュボード” に変更すると、OmniCRM の

例

ダッシュボード URL を `http://your-server/grafana/d/system-overview-123/system-metrics`

- ダッシュボード ID `system-overview-123`
- ダッシュボード名 “ダッシュボード”

例

- ID を変更します
- ダッシュボード名を変更します
- ダッシュボード名 `grafana_access` を変更します
- ダッシュボード UI

ダッシュボード URL

```
# ダッシュボード URL
VITE_FAQS_URL=https://support.yourcompany.com/faqs
VITE_SUPPORT_URL=https://support.yourcompany.com
```

ダッシュボード UI を変更します

例

```
# ダッシュボード yes | no
VITE_ALLOW_SOCIAL_LOGINS=yes
```

111

- **yes** - Google Facebook
- **no** -

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

55555

```
# Google Analytics 4 & Google Tag Manager
# GA4
# GTM
VITE GOOGLE TAG ID=G-XXXXXXXXXX
```

Google Analytics 4 (GA4) Google Tag Manager (GTM)

□ □ □ □ □

- **GA4** ID G-XXXXXXXXXX - Google Analytics 4
- **GTM** ID GTM-XXXXXXX - Google Tag Manager
- 其他跟踪代码

□□□□□

□□□□□ ID □□□□□□□□□□

- **G-** ID Google Analytics 4
- **GTM-** ID Google Tag Manager
- **XXXXXXXXXX**

100

- 1000000 - 1000000000000000
- 1000000000 - 1000000000000000000
- 1000000000 - 1000000000000000000 UI

□ □ □ □ □

1. 設定する ID

Google Analytics 4

- Google Analytics
- GA4
- Admin → Data Streams
- Web
- Measurement ID: G-XXXXXXXXXX

Google Tag Manager

- Google Tag Manager
-
- Container ID: GTM-XXXXXXX

2. .env 設定

```
VITE_GOOGLE_TAG_ID=G-XXXXXXXXXX
```

3. UI

```
cd OmniCRM-UI
npm run build
```

4. 実行

-
-
- Google Analytics 埋め込み

実行 - 実行

実行 - 実行

```
import { trackEvent } from './utils/googleAnalytics';

// 购物车
const handleCheckout = () => {
  trackEvent('checkout_initiated', {
    cart_value: 99.99,
    item_count: 3,
    page: '/cart'
  });
  // 结账
};
```

购物车 - 结账

```
import { trackEvent } from './utils/googleAnalytics';

const handleFormSubmit = (formType) => {
  trackEvent('form_submission', {
    form_type: formType,
    timestamp: new Date().toISOString()
  });
};
```

购物车 - 结账

```
import { trackPageView } from './utils/googleAnalytics';

// 页面视图
trackPageView('/virtual/page-path', 'Page Title');
```

购物车

関数	パラメータ	戻り値
<code>initializeGoogleTracking()</code>	初期化 関数	void
<code>trackEvent(eventName, eventParams)</code>	イベント 追跡関数	void
<code>trackPageView(path, title)</code>	ページビュー 追跡関数	void

注意事項

Google Analytics 利用規約を必ずご確認ください

- **GDPR**対応 - 欧州連合 (EU) 域内の個人データを適切に処理するための措置を講ずる
- **CCPA**対応 - カリフォルニア州の消費者プライバシー法に準拠する
- 同意 - ユーザーが Google Analytics を利用することに同意していることを確認する
- **Cookie** 管理 - Cookie を適切に管理し、ユーザーのプライバシーを保護する
- 匿名化 - Google Analytics の匿名化機能を利用する

インストール

ステップ	作業内容	コマンド
GA トラッキング ID を取得	トラッキング ID を取得	トラッキング ID は G-XXXXXXXXXX の形式で GTM-XXXXXXXX の形式で提供されます
環境変数設定	環境変数 ID を設定	プロジェクトの .env ファイルに追加します
ビルド	UI をビルドするための .env ファイルを設定	ターミナルで <code>npm run build</code> を実行して UI をビルドします
デプロイ	デプロイ環境を設定	デプロイ環境を設定します
確認	GA トラッキングを確認	トラッキングを確認します

完了

- 页面大小要尽可能小
- 页面大小 < 50KB 为佳
- 页面大小要尽可能小
- 在 `VITE_GOOGLE_TAG_ID` 中配置

配置

- 在 `OmniCRM-UI/src/utils/googleAnalytics.js`
- 在 `OmniCRM-UI/src/utils/usePageTracking.js`
- 在 `App.js` 中 `index.js` 中配置

配置

`VITE_TOPUP_PRICE_PER_DAY=10`

在 `OmniCRM-UI/src/constant/constant.js`

在

- `VITE_TOPUP_PRICE_PER_DAY` - 配置

在 `OmniCRM-UI/src/constant/constant.js` 中配置 `10` 美元 `USD` 为 `$10`

在 `OmniCRM-UI/src/constant/constant.js` 中配置 `features_topup_recharge`

配置

OmniCRM 配置

配置

在 `/OmniCRM-UI/src/assets/images/omnitouch/` 中配置

配置

- `DefaultLogoDark.png` - 다크 테마 로고 이미지
- `DefaultLogoLight.png` - 라이트 테마 로고 이미지

사용자 로고 이미지

- `logoSm.png` - 사용자 로고 이미지 (소형) 100x100px
- `logoDark.png` - 사용자 로고 이미지 (다크 테마) 200x50px
- `logoLight.png` - 사용자 로고 이미지 (라이트 테마) 200x50px

로고 이미지

로고 이미지 (소형) 100x100px

```
// src/Header.js
const tryImport = (filename) => {
  try {
    return require(`../assets/images/omnitouch/${filename}`);
  } catch (err) {
    return null; // 이미지 없음
  }
};

const userLogoSm = tryImport("logoSm.png");
const userLogoDark = tryImport("logoDark.png");
const userLogoLight = tryImport("logoLight.png");
```

로고 이미지

- **logoSm.png** - 사용자 로고 이미지 (소형)
- **logoDark.png** - 사용자 로고 이미지 (다크 테마)
- **logoLight.png** - 사용자 로고 이미지 (라이트 테마)

로고 이미지

1. 로고 이미지

- PNG 이미지
- 로고 이미지
- 로고 이미지 (소형)

2. 📁 OmniCRM

```
# 📁📁📁📁📁 omnitouch 📁📁  
cp /path/to/your/logoSm.png OmniCRM-  
UI/src/assets/images/omnitouch/  
cp /path/to/your/logoDark.png OmniCRM-  
UI/src/assets/images/omnitouch/  
cp /path/to/your/logoLight.png OmniCRM-  
UI/src/assets/images/omnitouch/
```

3. 📁 UI

```
cd OmniCRM-UI  
npm run build
```

4. 📁📁📁

- 📁📁📁📁📁📁📁 logoDark.png
- 📁📁📁📁📁📁📁 logoLight.png
- 📁📁📁📁📁📁📁 logoSm.png
- 📁📁📁📁📁📁📁 logoLight.png

📁📁📁📁📁📁

- 📁📁 - 📁📁📁📁📁📁📁📁📁📁📁📁📁📁
- 📁📁 - 📁📁📁📁📁📁📁📁📁📁
- 📁📁 - 📁📁📁📁📁📁 PNG
- 📁📁 - 📁📁 DPI 📁📁📁📁 2 📁📁📁
- 📁📁 - 📁📁📁📁📁📁📁📁📁📁📁📁📁

📁📁📁📁📁📁📁📁📁

📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁📁

📁📁 /OmniCRM-UI/src/pages/AuthenticationInner/authCarousel.js

📁📁📁📁

```
import logoLight from "../../assets/images/logo-light.png";

// ████████████

<!-- ![Company Logo](../absolute/path/to/your-logo.png) -->
    <h1>INVOICE</h1>
</div>

<div class="invoice-details">
    <p><strong>Invoice Number:</strong> {{ invoice_number }}</p>

    <p><strong>Date:</strong> {{ date }}</p>
    <p><strong>Due Date:</strong> {{ due_date }}</p>
    <p><strong>Billing Period:</strong> {{ start_date }} to {{ end_date }}</p>
</div>

<div class="customer-details">
    <h3>Bill To:</h3>
    <p>{{ client.name }}</p>
    <p>{{ client.address.address_line_1 }}</p>
    <p>{{ client.address.city }}, {{ client.address.state }}
    {{ client.address.zip_code }}</p>
    <p>{{ client.address.country }}</p>
</div>

<table>
    <thead>
        <tr>
            <th>Description</th>
            <th>Date</th>
            <th>Amount</th>
        </tr>
    </thead>
    <tbody>
        {% for transaction in transaction_list[0] %}
        <tr>
            <td>{{ transaction.title }}</td>
            <td>{{ transaction.created }}</td>
            <td>${{ "%.2f"|format(transaction.retail_cost) }}</td>
        </tr>
        {% endfor %}
    </tbody>
```

```

</table>

<div class="total">
    <p>Total Amount Due: ${{ "%.2f"|format(total_amount) }}
</p>
</div>

{% if paid %}
    <div style="text-align: center; color: green; font-weight:
bold;">
        PAID
    </div>
{% endif %}

{% if void %}
    <div style="text-align: center; color: red; font-weight:
bold;">
        VOID
    </div>
{% endif %}
</body>
</html>

```

📁📁📁📁📁

- 📁📁📁📁📁📁 - `file:///absolute/path/to/image.png`
- 📁 **CSS** - WeasyPrint 📁📁📁📁📁📁📁
- 📁📁📁📁📁📁 - 📁 `invoice_templates/rendered/` 📁 HTML
- 📁📁📁 - 📁 `<div style="page-break-after: always;"></div>` 📁📁📁📁
- 📁📁📁📁 - 📁 `@page` CSS 📁📁📁📁
- 📁📁📁📁 - 📁 Jinja2 🔪🔪🔪📁📁 `{{ "%.2f"|format(amount) }}`

📁📁📁📁📁📁📁

1. 📁📁📁📁📁

```

cd /OmniCRM/OmniCRM-API/invoice_templates
cp norfone_invoice_template.html
yourcompany_invoice_template.html

```

2. 設定

- 設定ファイルの作成
- 設定ファイルの読み込み
- 設定ファイルのバックアップ

3. 設定

例 `crm_config.yaml`

```
invoice:
  template_filename: 'yourcompany_invoice_template.html'
```

4. 実行

- CRM 実行
- PDF 生成
- 例 `invoice_templates/rendered/{invoice_id}.html` 生成

5. テスト

実行結果の確認

```
-- 例: PDF生成の確認
DELETE FROM Invoice_PDF_Cache;
```

PDF 生成

例: OmniCRM 実行 PDF

実行

- 実行 - PDF生成
- 実行 - PDF生成
- 実行 - PDF生成
- 実行 - 30 分間隔

🔒🔒🔒🔒

- Base64 🔒🔒 PDF 🔒🔒 Invoice_PDF_Cache 🔒🔒
- SHA256 🔒🔒🔒🔒🔒🔒🔒🔒
- 🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒

🔒🔒🔒🔒🔒🔒

```
# 📄 OmniCRM API 📄 Python shell 📄  
from services.invoice_service import cleanup_old_pdf_cache,  
invalidate_invoice_cache  
from utils.db_helpers import get_db_session  
  
session = get_db_session()  
  
# 🔒🔒🔒 30 🔒🔒🔒  
result = cleanup_old_pdf_cache(session, days_old=30)  
print(result) # {'status': 'success', 'deleted_count': 15}  
  
# 🔒🔒🔒🔒🔒🔒  
invalidate_invoice_cache(session, invoice_id='12345')
```

API 🔒🔒

🔒🔒/🔒🔒🔒🔒 PDF🔒

GET /invoice/pdf/{invoice_id}

🔒🔒 PDF 🔒🔒🔒🔒🔒🔒🔒 Stripe 🔒🔒🔒🔒

🔒🔒🔒🔒

- 🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒
- 🔒🔒🔒🔒🔒🔒
- 🔒🔒🔒/🔒🔒🔒🔒🔒🔒🔒🔒

部署指南

API (crm_config.yaml)

1. 进入 `OmniCRM-API/crm_config.yaml`
2. 配置
3. 重启 API 服务

```
cd OmniCRM-API
sudo systemctl restart omnicrm-api
# 重启 API 服务
./restart_api.sh
```

部署完成

UI (.env)

1. 进入 `OmniCRM-UI/.env`
2. 配置
3. 重启 UI 服务

```
cd OmniCRM-UI
npm run build
```

4. 重启 UI 服务 Web 服务

部署完成

运行 `npm start` 启动服务

環境構築

前提

- 開発環境 - 開発環境構築用 `.gitignore`
- ネットワーク - 約 16 台のサーバーを接続
- 認証 - 認証サーバー
- ログ管理 - 約 IP 管理サーバー
- 監視 - ログ/メトリクス/アラート
- 約 **API** サービス - 開発環境
- 約 **IP** サービス - 約 API サービス

インストール

- ネットワーク - 開発環境構築
- 認証 - 開発環境構築
- ログ管理 - 開発環境構築
- ネットワーク - 約 git サービス

デプロイ

- ネットワーク - 開発環境構築
- ネットワーク - 開発環境構築 OCS サービス
- 約 **Grafana** - 開発環境構築

テスト

- ネットワーク - 約 UI サービス
 - ネットワーク - 開発環境構築   サービス
 - ネットワーク - 開発環境構築
 - ネットワーク - 開発環境構築
-

UI

UI

UI

- UI UI
- UI API/UI

YAML

- YAML
- YAML `yamllint crm_config.yaml`

-
-

Stripe

-
- `publishable_key`

- Mailjet ID
- Mailjet API / ID

PDF

- WeasyPrint `pip install weasyprint`
- `crm_config.yaml`
- `invoice_templates/rendered/` HTML
- API Jinja2

PDF

- 在代码中指定 `file:///full/path/to/image.png`
- 使用绝对路径
- 使用 `data-uri` 嵌入 PNG 或 JPEG 数据
- 使用 `data-uri` 嵌入 SVG 数据

布局

- 使用 CSS 布局模型
- 使用 CSS 布局模型 `flexbox` 或 `grid` 布局
- 使用 `flexbox` 布局
- 使用 `flexbox` 布局 `div` 容器

缓存

- 在代码中指定 `invalidate_invoice_cache()`
- 使用 `invalidate_invoice_cache()`
- 在代码中指定 `Invoice_PDF_Cache` 缓存

部署

部署 OmniCRM 到生产环境

部署

- ☐ 在 `.env.example` 文件中指定 `.env`
- ☐ 使用 `docker-compose`
- ☐ 使用 `CGRates` 容器
- ☐ 在 `crm_config.yaml` 文件中指定配置
 - ☐ 数据库
 - ☐ 邮件
 - ☐ Mailjet API 密钥 ID
 - ☐ 邮件模板
 - ☐ 邮件发送
 - ☐ CRM 的 URL

- ☐ OCS/CGRates
- ☐ SMSC
- ☐ JWT
- ☐ Stripe
- ☐ API IP

Preparation

- ☐ Copy `OmniCRM-UI/.env.example` to `OmniCRM-UI/.env`
- ☐ Get Google Maps API
- ☐ Get Stripe
- ☐ Get Images
 - ☐ Images
 - ☐ Images
 - ☐ Images
 - ☐ Images
- ☐ Get Fonts
 - ☐ Fonts
 - ☐ Fonts
 - ☐ Fonts
 - ☐ Fonts
- ☐ Get Web
- ☐ Get URL
- ☐ Get Google Analytics/Tag Manager ID

Deployment

- ☐ Copy `logoSm.png`, `logoDark.png`, `logoLight.png`
- ☐ Copy `OmniCRM-UI/src/assets/images/omnitouch/`
- ☐ Copy `HTML`
- ☐ Copy `OmniCRM-API/invoice_templates/`
- ☐ Copy `crm_config.yaml`
- ☐ Copy PDF

- ☐ UI `npm run build`

UI

- ☐ Authentication
- ☐ JWT
- ☐ Stripe
- ☐ Mailjet API
- ☐ Mailgun
- ☐ API IP
- ☐ SSL/TLS
- ☐ HTTPS
- ☐ CORS
- ☐ OAuth
- ☐ OAuth2

API

- ☐ Authentication
- ☐ Authentication
- ☐ Authentication
- ☐ PDF
- ☐ Stripe
- ☐ 2FA
- ☐ OAuth
- ☐ OCS
- ☐ ActionPlan
- ☐ OAuth

API

- ☐ Docker
- ☐ MySQL/PostgreSQL
- ☐ CGRates

- ☐ OmniCRM API
 - ☐ OmniCRM UI
 - ☐ nginx traefik
 - ☐ Grafana Prometheus
 - ☐
 - ☐
 - ☐
 - ☐
-

API

- RBAC
-
- Ansible
-
-
-
-
-
- administration_api_keys - API
- integrations_mailjet -
- concepts_api - API

OmniCRM

OmniCRM

IP GPON ONT CPE

Customer Attributes <administration_attributes>
Customer Tags <administration_tags>

CPE

inventory_items_list
UI Ansible Complete Product Lifecycle Guide - Inventory Requirements <guide_product_lifecycle>

OmniCRM

1. SIM
- 2.
- 3.
4. GPON CPE

SIM□□□

Omnitouch 1000 SIM SIM SIM SIM SIM SIM SIM SIM SIM
SIM

SIM卡已插入，但无法识别。请检查SIM卡是否插紧，或尝试重新插入SIM卡。

❗❗❗ 请勿将手机放入水中，以免损坏设备。

XXXXXXXXXXXXXXXXXXXXSIMXXXXXXXXXXXXXXXXXXXXSIMXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXSIMXXXXXXXXXXXX

GPON□□□□□□

GPON

[illegible]

□ □

OmniCRM

--	--	--	--

InventoryTemplate[]

- MAC itemtext1 itemtext2
-
- itemtext1 ? itemtext2
- IP

20 itemtext1 itemtext20

```

itemtext1_label itemtext20_label
itemtext1_label ""MAC"" itemtext2_label """"

```

Inventory Templates

- O □□□□□□□□□□□□□□□□□□□□□□□□

11

[illegible]

□□□□□

- [illegible]

□ □ □ □ □ □

- ☐ □□□□□□□□□□□□□□□□
- ☐ □□□□□□□□□□□□/□□□□□□□□□□

4.

□ □ □ □ □ □

□ □ □ □ □ □ □ □

1. $\square\square\square\square\square\square \rightarrow \square\square$
2. $\square\square\square\square\square\square\square\square$
3. $\square\square\square\square\square\square$
4. $\square\square\square\square\square\square\square\square$
5. $\square\square\square\square$

11

```
00000000 itemtext1_label 00000000000000000000000000000000
```

11

```
inventory_items_list
```

□□□□□□

SIM□□□□

- □□□“SIM□”
- □□□□□2.50
- □□□□□10.00
- □□□□1□□□“ICCID”
- □□□□2□□□“IMSI”
- □□□□3□□□“SIM□□”□□□/□□□□□

□□□□□□□□

- □□“□□□□”
- □□□□□1.00
- □□□□□0.00
- □□□□1□□□“□□□□”
- □□□□2□□□“□□□□”
- □□□□3□□□“□□□□”□□□/□□□□□□

□□□□**CPE**□□□□

- □□□“□□□□CPE”
- □□□□□250.00
- □□□□□450.00
- □□□□1□□□“MAC□□”
- □□□□2□□□“□□□□”
- □□□□3□□□“□□□□□”
- □□□□4□□□“□□□□”
- □□□□5□□□“□□□□”

GPON ONT□□□□

- □□□“GPON ONT”
- □□□□□45.00
- □□□□□0.00□□□□□□□□□□

- 10001000“000”
- 10002000“MAC00”
- 10003000“PON00”
- 10004000“00”

--	--	--	--	--	--	--	--	--

InventoryTemplate

- [illegible]

OmniCRM

Ansible



項目

項目名、項目ID、項目説明、項目単位

項目単位

- **inventory_id** - 項目ID
- **item** - 項目名“SIM”“項目”“項目”
- **inventory_template_id** - 項目ID
- **customer_id** - 項目ID
- **service_id** - 項目ID
- **item_location** - 項目ID
 - 項目ID
 - 項目ID
- **item_state** - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID
 - 項目ID

項目単位

項目単位20項目単位

- **itemtext1** - 項目ID **itemtext1_label** 項目ID
 - 項目ID“MAC”
 - 項目ID“SIM”“ICCID”
 - 項目ID“項目”
- **itemtext2** **itemtext20** - 項目ID
 - 項目ID“項目”
 - 項目ID“IMSI”

- `itemtext3` “ ”

`itemtext1_label` `itemtext2_label`

- **wholesale_cost** - /
- **retail_cost** -
- **sold_date** -

CPE ONT

- **address_line_1** -
- **address_line_2** -
- **city** -
- **state** -
- **zip_code** - /
- **country** -

Web UI

Web UI

- **google_maps_place_id** - Google ID
- **plus_code** - Google
- **latitude** -
- **longitude** -

-
-
-
-

管理URL

`management_url` 管理URL

- **management_url** - 管理URL
 - WebURL `https://192.168.1.1` ONT CPE
 - **eSIM** eSIM LPA `LPA:1$smgp.example.com$ACTIVATION-CODE-HERE`
 - 管理URL

管理

`management_url` 管理URL

- 128x128 URL
- 64x64
-

管理

- IP
- **eSIM** CRM eSIM
-

- **management_username** - 管理
- **management_password** - 管理

管理

管理

- **config_content** - 配置
 -
 - CPE
- **config_file_path** - 配置
 -
 -

Inventory

- **inventory_notes** - Inventory notes
 - Item
 - Location
 - Status
 - Quantity
 - Date
- **created** - Date and time when the item was created
- **last_modified** - Date and time when the item was last modified

Item

Item 1: Physical SIM

```
{
  "inventory_id": 1001,
  "item": "SIM Card",
  "inventory_template_id": 5,
  "itemtext1": "8961234567890123456",
  "itemtext2": "310120123456789",
  "itemtext3": "Physical",
  "item_location": "Warehouse A, Shelf 3",
  "item_state": "Assigned",
  "customer_id": 456,
  "service_id": 789,
  "wholesale_cost": 2.50,
  "retail_cost": 10.00,
  "sold_date": "2025-01-15T10:30:00Z",
  "inventory_notes": "Activated on 2025-01-15"
}
```

Item 2: eSIM

```
{
  "inventory_id": 1002,
  "item": "eSIM",
  "inventory_template_id": 6,
  "itemtext1": "8961234567890123457",
  "itemtext2": "310120123456790",
  "itemtext3": "eSIM",
  "item_location": "Virtual Inventory",
  "item_state": "Assigned",
  "customer_id": 457,
  "service_id": 790,
  "management_url": "LPA:1$smdp.example.com$ACTIVATION-CODE-ABC123XYZ",
  "wholesale_cost": 0.00,
  "retail_cost": 0.00,
  "sold_date": "2025-01-16T14:20:00Z",
  "inventory_notes": "eSIM profile ready for activation"
}
```

eSIM UI LPA eSIM

3 CPE -

```
{
  "inventory_id": 2001,
  "item": "Fixed Wireless CPE",
  "inventory_template_id": 10,
  "itemtext1": "AA:BB:CC:DD:EE:FF",
  "itemtext2": "FW2024-12345",
  "itemtext3": "v2.4.1",
  "itemtext4": "Ubiquiti",
  "itemtext5": "LiteBeam AC Gen2",
  "item_location": "Customer Site",
  "item_state": "Assigned",
  "customer_id": 789,
  "service_id": 1234,
  "address_line_1": "123 Main Street",
  "address_line_2": "Apt 4B",
  "city": "Sydney",
  "state": "NSW",
  "zip_code": "2000",
  "country": "Australia",
  "latitude": "-33.8688",
  "longitude": "151.2093",
  "management_url": "https://192.168.100.1",
  "management_username": "admin",
  "management_password": "encrypted_password_here",
  "config_file_path": "/configs/cpe/fw2024-12345.conf",
  "inventory_notes": "Installed 2025-01-10. Customer reports
excellent signal. Pointing: Azimuth 45°, Elevation 15°"
}
```

4 GPON ONT

- 000000000340000000000000000000000000IP00000050000000000000

```
{
  "inventory_id": 4001,
  "item": "Phone Number",
  "inventory_template_id": 20,
  "itemtext1": "+61412345678",
  "itemtext2": "Melbourne",
  "itemtext3": "Mobile",
  "item_location": "Australia - VIC",
  "item_state": "Assigned",
  "customer_id": 555,
  "service_id": 888,
  "wholesale_cost": 1.00,
  "retail_cost": 0.00,
  "inventory_notes": "Ported from Telstra on 2025-01-05"
}
```

0000000000

item_state0000000000000000

- 0 → 000 - 0000000000
- 000 → 000 - 0000000000000000
- 0 → 0000 - 0000000000000000
- 000 → 00 - 00000000RMA0000
- 0000 → 00 - 00000000000000
- 00000 → 00 - 00000000000000
- 00/000 → 0 - 0000000000

00000000000000000000

- 00000000000000
- 000000000000000000
- 000000000000000000
- 0000000000000000
- 0000000000000000

-  ☐
- ☐
- ☐
- ☐

☐

☐

1. ☐
2. ☐ ☐ ☐
3. ☐ ☐ ☐
4. ☐
 - ☐
 - ☐
 - ☐ URL
 - ☐
 - ☐ 2099-01-01
5. ☐
6. ☐ ☐

API 1

API 1 1.1

1. API 1.1.1
2. API 1.1.2
3. API 1.1.3
4. API 1.1.4
5. API 1.1.5
6. API 1.1.6

API 2

API 2 2.1

1. API 2.1.1
2. API 2.1.2
3. API 2.1.3
4. API 2.1.4
5. API 2.1.5

API 3

API 4

API 4 4.1 API 4.1.1

API 4.1.2

```
PUT /crm/tag/
{
  "tag_text": "VIP Customer",
  "tag_hex_color": "FFD700",
  "tag_link": "https://example.com/vip",
  "tag_active_date": "2025-01-01 00:00:00",
  "tag_deactivate_date": "2099-12-31 23:59:59",
  "customer_id": 12
}
```

□□□□□

```
PATCH /crm/tag/tag_id/{tag_id}
{
  "tag_text": "Updated Tag Text",
  "tag_hex_color": "FF0000"
}
```

□□□□□□□□

```
GET /crm/tag/customer_id/{customer_id}
```

□□□□□

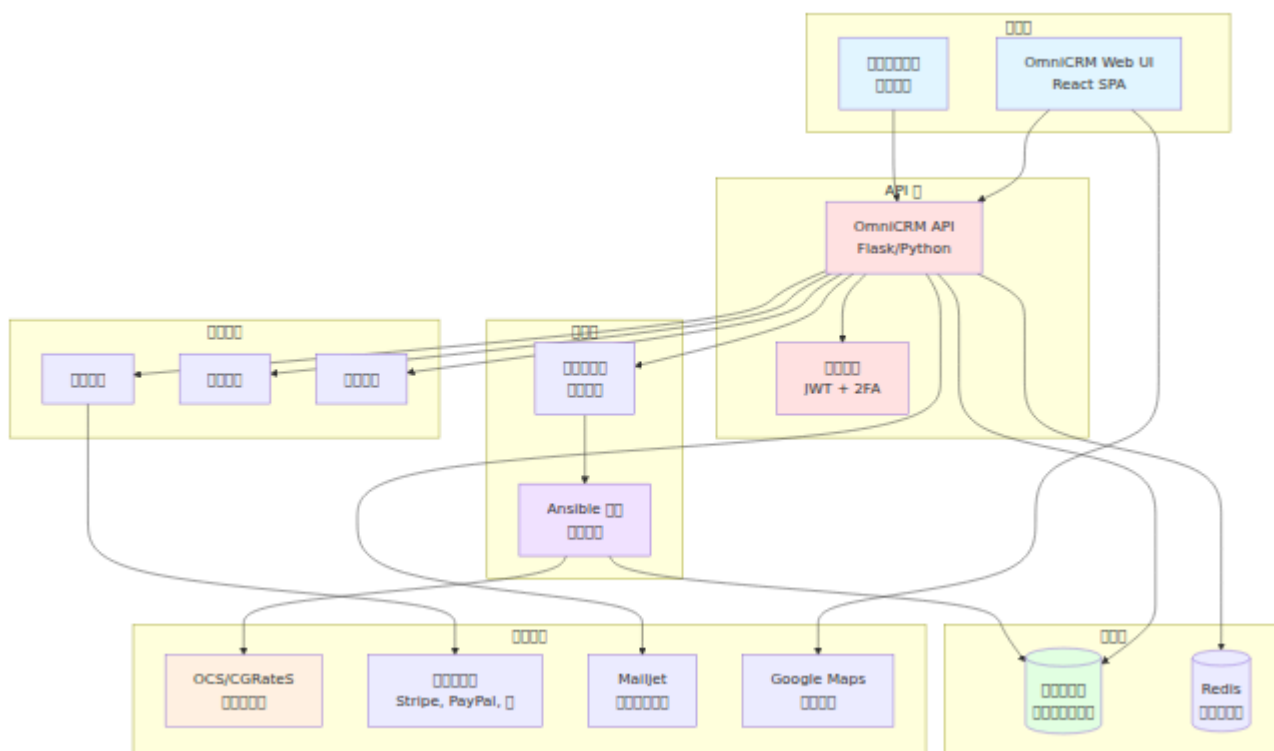
```
DELETE /crm/tag/tag_id/{tag_id}
```

OmniCRM アプリケーション

このドキュメントは OmniCRM アプリケーションのアーキテクチャを説明します。

概要

OmniCRM アプリケーションは BSS/OSS アプリケーションのアーキテクチャを説明します。



構成

1. 環境

OmniCRM Web UI

- React アプリケーション
- 静的コンテンツのホスティング
- 静的コンテンツの配信
- 静的コンテンツのキャッシュ

□□□□□

- □□□□□□
- □□□□□□□□
- □□□□□□
- □□□ UI □□□□□□□□□□

2. API □

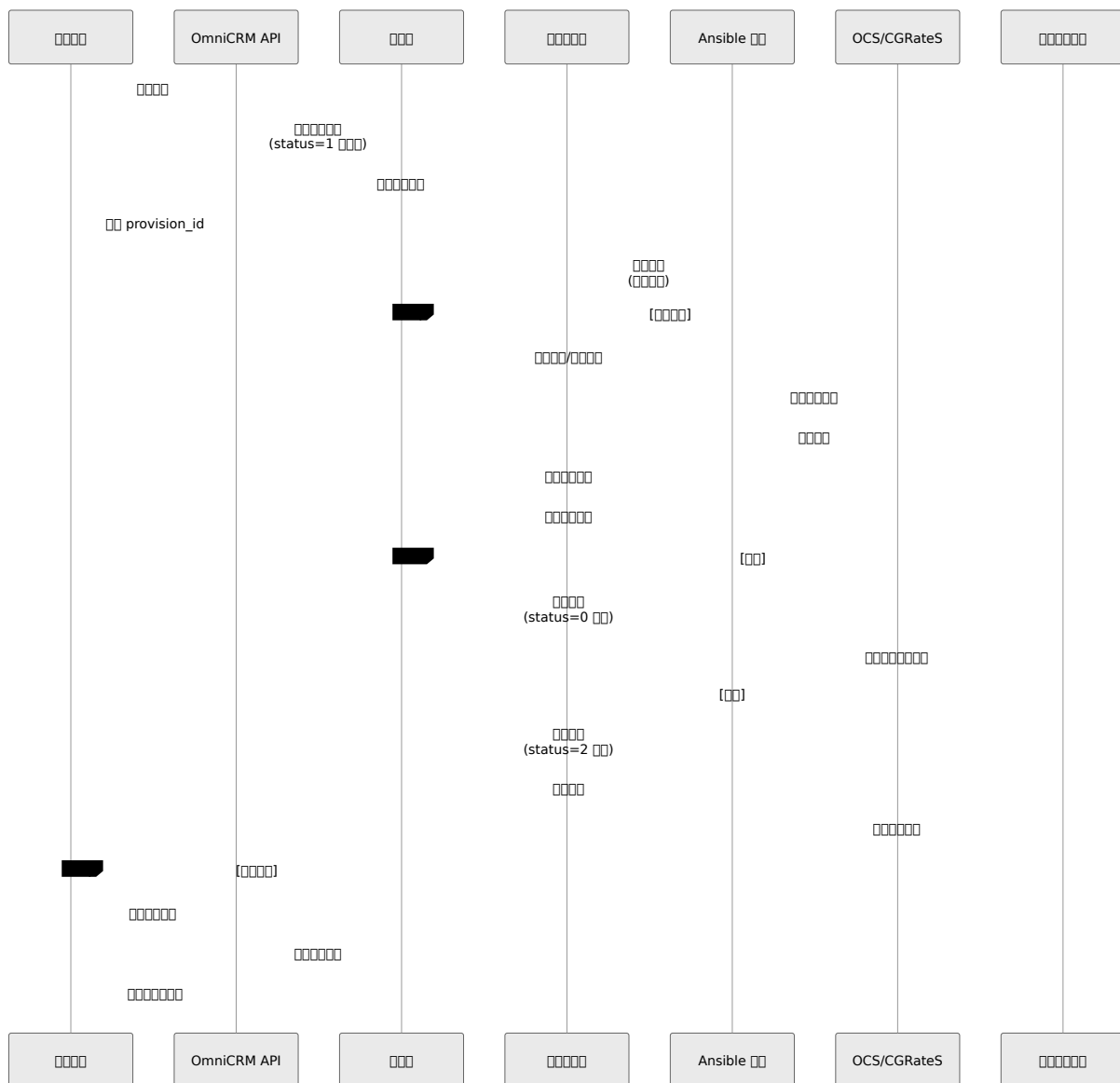
OmniCRM API (Flask/Python)

- □□□□□ RESTful API
- OpenAPI/Swagger □□
- □□ JWT □□□
- □□□□□□□
- WebSocket □□□□□□

□□ **API** □□:

- `/crm/customer/*` - □□□□
- `/crm/service/*` - □□□□
- `/crm/product/*` - □□□□
- `/crm/provision/*` - □□□□
- `/crm/transaction/*` - □□□□
- `/crm/invoice/*` - □□□□

3.

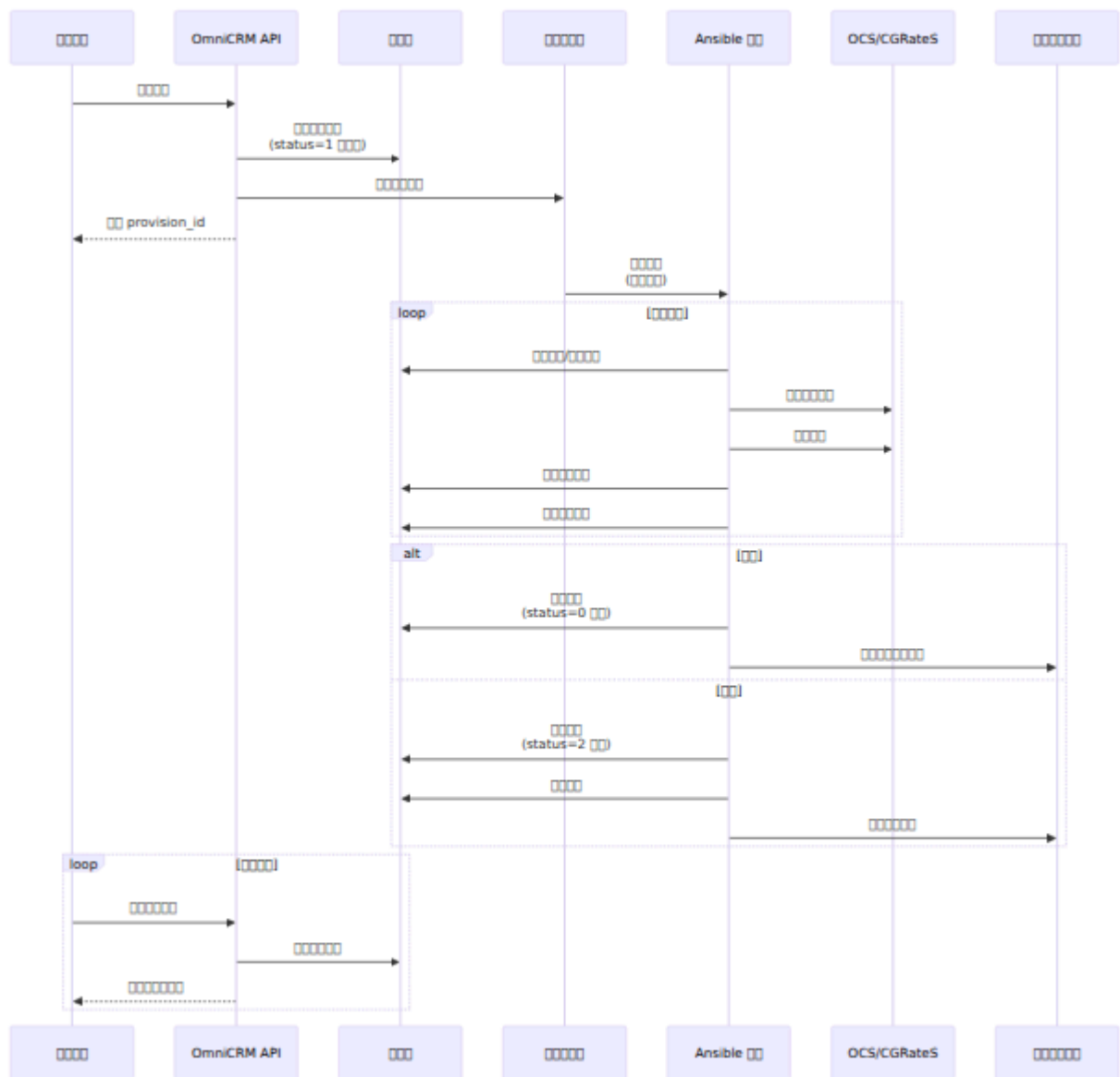


4.

OCS/CGRateS ☐

- 
- 
- 
- 
- 

00000



5. 测试

OmniCRM 测试用例如下表所示

测试用例

测试 - 测试用例

欄名	属性	説明
customer_id	整数 (PK)	顧客ID
customer_name	文字列	顧客名
customer_account_type	整数	'個人' 或 '法人'
customer_status	整数	'有効', '無効', '凍結', '解約'
customer_payment_type	整数	'現金' 或 'クレジット'
customer_enabled	整数	有効フラグ
tax_identifier	文字列	VAT/GST 番号
contract_start_date	日付	契約開始日
contract_end_date	日付	契約終了日

備考 - 詳細情報は別表参照

欄名	データ型	コメント
contact_id	整数 (PK)	連絡先ID
customer_id	整数 (FK)	顧客ID
contact_firstname	文字列	姓
contact_lastname	文字列	名
contact_email	文字列	Eメール
contact_phone	文字列	電話番号
contact_types	文字列	'個人', '会社', 'その他'

住所 - 住所詳細

欄名	データ型	コメント
site_id	整数 (PK)	サイトID
customer_id	整数 (FK)	顧客ID
site_name	文字列	サイト名
address_line_1	文字列	住所1行目
city, state, zip_code	文字列	住所2行目
latitude, longitude	浮点	GPS 緯度経度
google_maps_place_id	文字列	Google Maps 場所ID
plus_code	文字列	Google Plus Code

住所詳細

表 - 表头

表	表	表
service_id	表 (PK)	表
customer_id	表 (FK)	表
product_id	表 (FK)	表
site_id	表 (FK)	表
service_name	表	表
service_uuid	表	表
service_status	表	表
service_billed	表	表
wholesale_cost	表	表
retail_cost	表	表
bundled_parent	表 (FK)	表

表 - 表头

欄名	データ型	コメント
product_id	整数 (PK)	製品ID
product_name	文字列	製品名
product_slug	文字列	URL フレンドリー
category	整数	'1', '2', '3', '4'
provisioning_play	文字列	Ansible プレイブック
provisioning_json_vars	JSON	変数
inventory_items_list	文字列	インベントリ
retail_cost	小数	小売価格
retail_setup_cost	小数	小売設置価格
enabled	ブール値	有効/無効

備考

更新 - 2024/01

欄名	データ型	制約
transaction_id	NUMBER(10)	PK
customer_id	NUMBER(10)	FK
invoice_id	NUMBER(10)	FK
service_id	NUMBER(10)	FK
title	VARCHAR2(255)	
retail_cost	NUMBER(10,2)	
tax_percentage	NUMBER(5,2)	
tax_amount	NUMBER(10,2)	
void	NUMBER(1)	

注: 税額は別途計算

欄名	データ型	コメント
invoice_id	整数 (PK)	請求ID
customer_id	整数 (FK)	顧客ID
paid	ブール値	請求済み
void	ブール値	キャンセル済み
payment_reference	文字列	Stripe 請求 ID
start_date, end_date	日付	有効期間
due_date	日付	支払期限
retail_cost	小数	小売価格

備考

請求 - 請求明細

欄名	属性	説明
inventory_id	PK	在庫ID
customer_id	FK	顧客ID
service_id	FK	サービスID
inventory_template_id	FK	在庫テンプレートID
item	TEXT	品名SIM 品名
item_state	TEXT	'0', '1', '2' のいずれか
itemtext1-20	TEXT	品名
management_url	TEXT	管理URL
config_content	TEXT	設定内容

在庫 - 在庫テンプレート

欄名	属性	説明
inventory_template_id	PK	在庫テンプレートID
item	TEXT	品名
itemtext1_label	TEXT	itemtext1 品名
itemtext2_label	TEXT	itemtext2 品名

品名

品名 - 品名

欄名	制約	説明
provision_id	PK	プロビジョニングID
product_id	FK	製品ID
customer_id	FK	顧客ID
service_id	FK	サービスID
provisioning_play	Text	Ansible プレイブック
provisioning_status	Text	0=成功, 1=失敗, 2=中止

プロビジョニング - イベント

欄名	制約	説明
provision_event_id	PK	プロビジョニングイベントID
provision_id	FK	プロビジョニングID
event_name	Text	イベント名
event_number	Text	イベント番号
provisioning_status	Text	0=成功, 1=失敗, 2=中止
provisioning_result_json	JSON	プロビジョニング結果JSON

プロビジョニング結果

プロビジョニング - イベント

欄名	データ型	制約
id	整数 (PK)	一意
username	文字列	一意
email	文字列	一意
email_verified	ブール値	
is_2fa_enabled	ブール値	
totp_secret	文字列	TOTP 秘密鍵

ユーザ - ユーザー

欄名	データ型	制約
id	整数 (PK)	一意
name	文字列	一意
description	文字列	

ユーザ - ユーザー

欄名	データ型	制約
id	整数 (PK)	一意
name	文字列	一意
description	文字列	

ユーザ:

- 國際化國際化國際化
- 國際化國際化國際化
- 國際化國際化國際化國際化國際化國際化

國際化

Stripe 國際化

- 國際化國際化
- 國際 PCI 國際化
- 國際化國際化
- 國際化
- 國際化國際化

Mailjet 國際化國際化

- 國際化國際化國際化國際化國際化國際化
- 國際化國際化
- 國際化國際化
- 國際化國際化

Google Maps

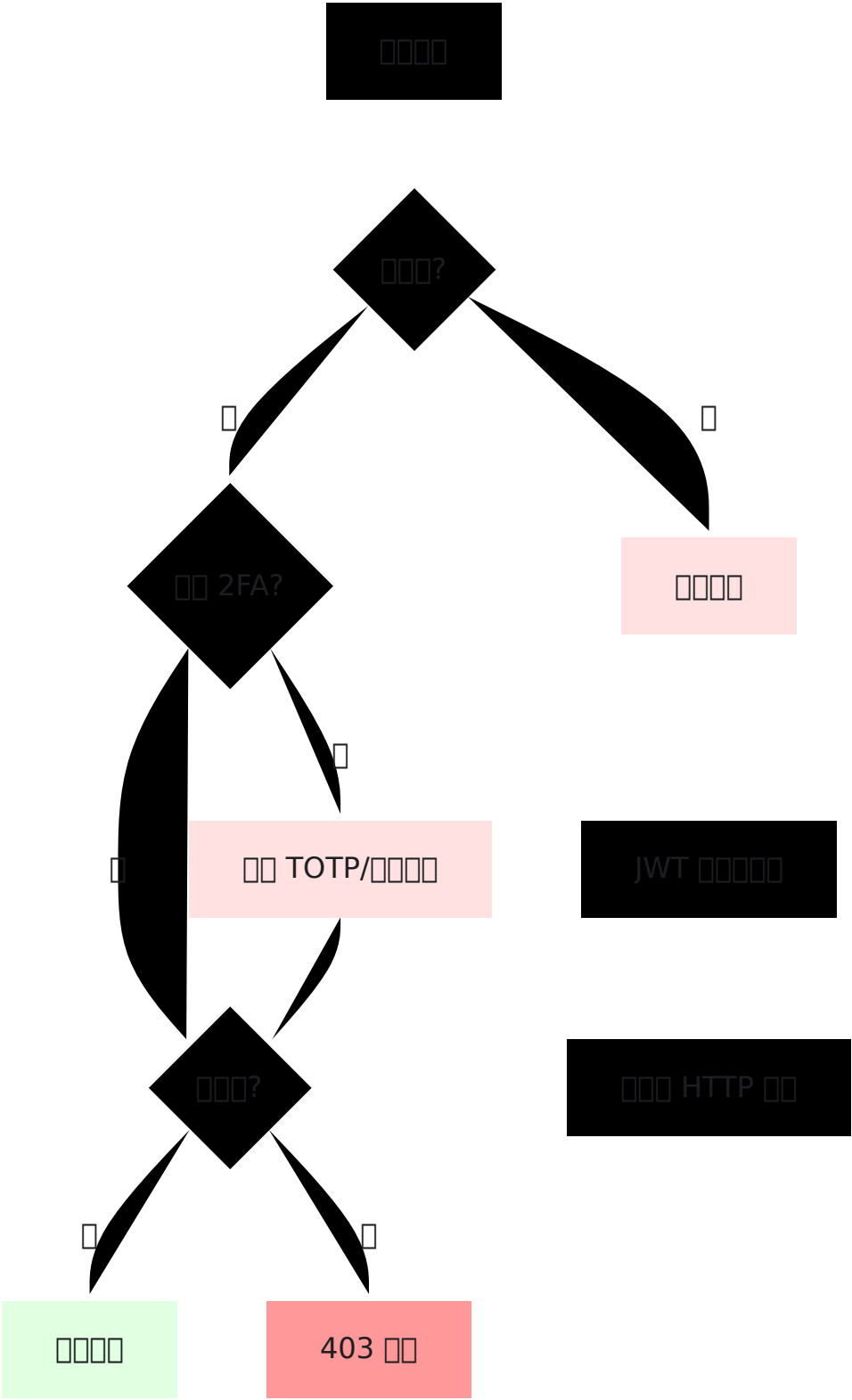
- 國際化國際化
- 國際化國際化國際化國際化
- Plus Code 國際化
- 國際化國際化

OCS/CGRateS

- 國際化國際化
- 國際化國際化
- 國際化國際化
- CDR 國際化

- 认证流程

认证流程

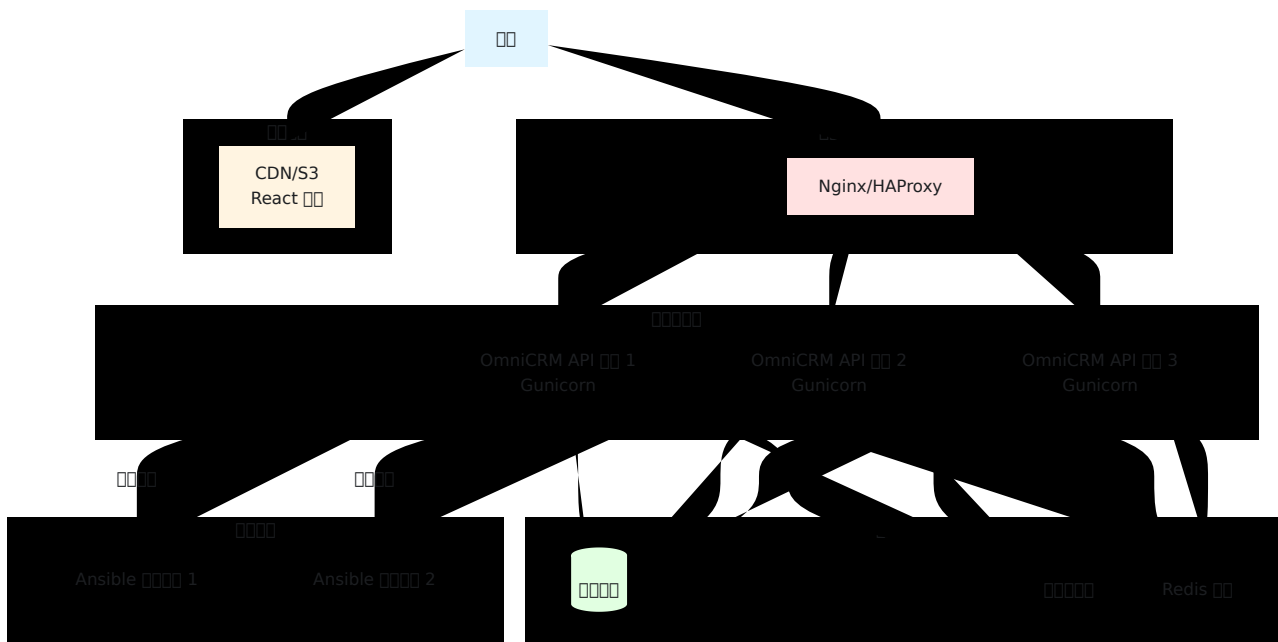


认证成功:

- 使用 JWT 认证
- 支持 TOTP 认证
- 支持 RBAC 权限控制
- 支持 Redis 缓存
- 使用 bcrypt 加密
- 支持 Docker 部署
- 支持 Kubernetes 部署

架构图

架构图：



技术栈

技术栈：

- Python 3.x
- Flask 框架
- SQLAlchemy ORM
- Alembic 迁移

- Ansible 配置管理

UI:

- React
- 路由
- React Router
- Axios 请求 API 数据

数据库:

- 数据库连接池管理
- Redis 缓存管理

第三方:

- CGRateS 计费/统计
- Stripe 支付
- Mailjet 邮件发送
- Google Maps 地图服务

部署与运维

部署:

- 部署 API 服务到容器
- 部署数据库
- Redis 部署到容器

监控:

- 数据库连接池监控
- 部署
- 部署到容器

部署:

- `ansible`
- `Ansible`
- `ansible`
- `ansible`

--	--	--	--	--	--	--

[illegible]

11

- API 開発
- 開発/テスト環境
- 開発環境
- 開発環境 OCS Stripe Mailjet
- 開発環境

□□□□ ■■

-  Flask
-  Ansible
- 
- 

URL: `/crm/metrics` Prometheus `scrape_configs`

1111

- **Ansible**  -   
- **GitLab**  -   
- **OpenStack**  -   
- **API**  - **API**   
- **RBAC**  -   

OmniCRM 2FA RBAC Self-Care Portal

OmniCRM 2FA RBAC Self-Care Portal (2FA) RBAC Self-Care Portal UI

Self-Care Portal <self_care_portal> RBAC <rbac>

OmniCRM

OmniCRM

- OmniCRM/2FA - OmniCRM
- OmniCRM (2FA) - OmniCRM TOTP OmniCRM
- OmniCRM - OmniCRM OmniCRM 30
- OmniCRM - OmniCRM OmniCRM
- OmniCRM - OmniCRM 2FA OmniCRM
- OmniCRM - OmniCRM GoogleAppleFacebook OmniCRM
- OmniCRM - OmniCRM OmniCRM

OmniCRM

OmniCRM OmniCRM

□□□□

□□□□□

1. □□ □□□□□□□□□□□□□□□□
2. □□ □□
3. □□□□ “□□□□ 30 □” □□□□□
4. □□ “□□”

□□□□□□□□□□

- □□ **2FA** □□□□□□□□□□□□□□□□
 - □□ → □□□□□□ (/self-care)
 - □□/□□□ → □□□□□□ (/customers)
 - **CBC** □□ → □□□□□□ (/create-cell-broadcast)
- □□ **2FA** □□□□□ 2FA □□□□□

□□□□□□

□□□□□□□□ **30** □□□□□□□□□□□□□□□□□□□□□□□□ HTTP-only cookies□

□□/□□□□□□

□□ □□□□ (□) □□□□□□□□□□

如何 2FA 设置

如何设置 2FA 验证/如何设置 2FA 验证

如何设置

1. 如何设置 Google Authenticator/AUTHY 验证
2. 如何 OmniCRM 验证
3. 如何 6 位验证
4. 如何 6 位验证
5. 如何设置验证

如何设置

如何设置

1. 如何 “验证” 验证
2. 如何设置验证 3fa5b9c2
3. 如何 “验证”
4. 如何设置验证

如何

如何 “验证” 验证

🔒 安全設定

🔗 `REACT_APP_ALLOW_SOCIAL_LOGINS=yes` を有効にする

📱 [Google ログイン] [● Apple ログイン] [Facebook ログイン]

🔑 2FA を有効にするには、このコードを入力してください

📄 2FA 設定

🔑 “2FA” を有効にする

🔒 (2FA) 設定

🔑 2FA を有効にするには、2FA の TOTP を入力してください

🔑 2FA 設定

🔑 2FA 設定

🔑 2FA 設定

🔑 2FA を有効にするには、2FA の TOTP を入力してください

🔑 1. 2FA 設定

🔑 2FA 設定

🔑 2FA 設定

🔑 2FA を有効にするには、2FA の TOTP を入力してください

🔑 2. 2FA 設定

🔑 2FA 設定

🔑 2FA 設定

1. 登入驗證器安裝

- iOS → Apple App Store → “Google Authenticator”
- Android → Google Play → “Google Authenticator”
- 電腦端 Authy → Microsoft Authenticator 1Password

2. 驗證碼

- 驗證碼
- 密碼 “+” 驗證碼
- 密碼 “驗證碼”
- 驗證碼
- 驗證碼 “OmniCRM” 密碼 6 位數

3. 驗證碼

- 密碼 驗證碼 8 位數
- 驗證碼
- 驗證碼
- 驗證碼
- 密碼 “驗證碼” 驗證碼

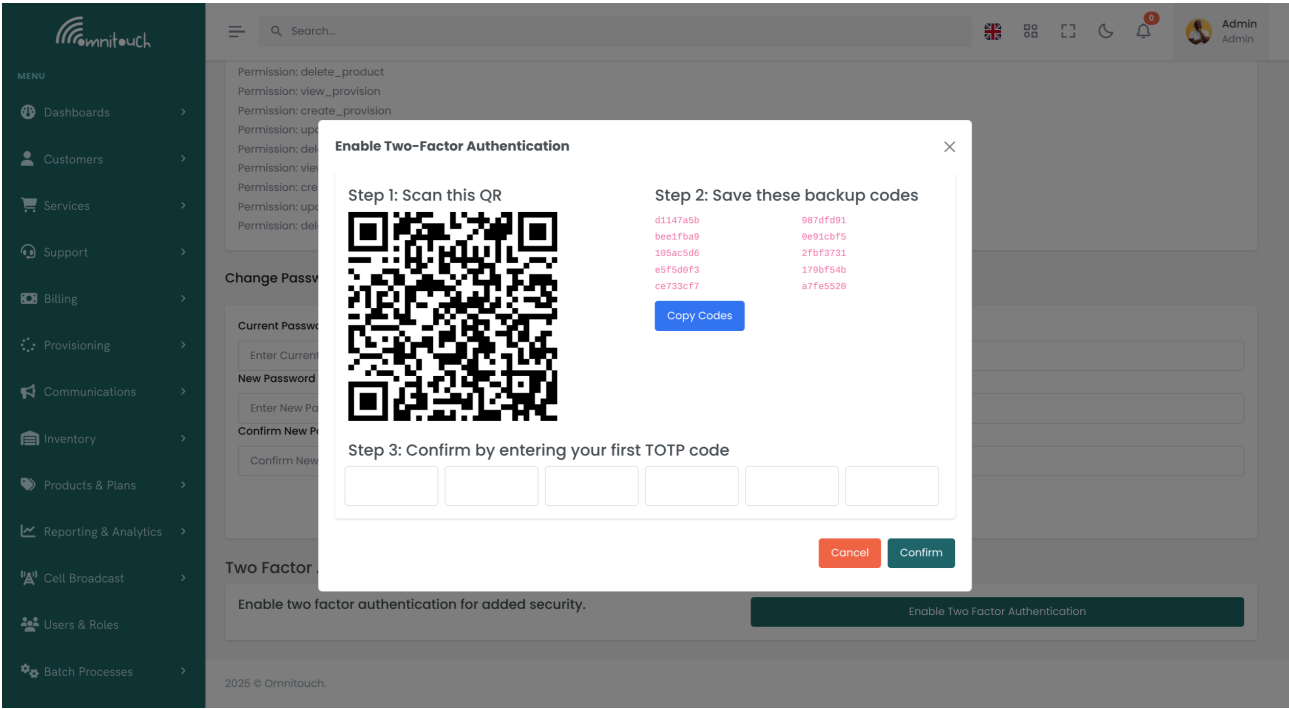
4. 驗證碼

- 驗證碼 驗證碼 6 位數
- 密碼 “密碼”
- 驗證碼 2FA 驗證碼

3 密碼 2FA 密碼

驗證碼

驗證碼 2FA 密碼



□□□□□□□□□□

□□□□□□□□□□□□□□□□

□ **1** □□□□□□□□

□□□□□□□□ “□□□□□□”

1. 如何 设置邮件
2. 如何 “发送邮件”

发送邮件

- 如何设置邮件发送频率
- 如何设置 Mailjet 发送邮件  频率
- 如何设置邮件发送频率为 1 次/天
- 如何设置 “发送邮件频率”

发送邮件频率

如何设置“发送邮件频率”

2 发送邮件

发送邮件频率

OmniCRM

<<https://yourcompany.com/reset-password/abc123token456>>

1

3

- 1.
- 2.
3. “”

- 8
-

- “”
-
-

XXXXXXXXXXXX

[XXXXXXXXXX]

XXXXXXXXXXXX

XXXXXXXXXXXX❖❖XXXXXXXXXXXX

XXXXXX

XXXXXXXXXXXX

XX XXXX XX XX John Smith <john@example.com> +44 123... ➡ XXX Jane
Doe <jane@example.com> +44 456... ➡ XX Bob Wilson
<bob@example.com> +44 789... ➡ XXX ☒

XXXXXX

- ➡ XX - XXXXXXXXXXXXXXX
- XX XX - XXXXXX
- XX XXXX - XXXXXX
- XX XX **2FA** - XXXXX 2FAXXXX 2FA XXXX
- ☒ XXXXXXX - XXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

1 XXXXXXXXXXXXX (X)

XXXXXXXXXX

XXXXXXXXXXXX

XXX John Smith (<john@example.com>)

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

[] []

2

“ ”

3

John Smith

[]

• • •

[]

-
-
-
-

- 1.
2. “ ”
- 3.
- 4.

2FA

1 2FA ()

2FA

00000000

2 000000

“2FA”

3 0000

0000

John Smith 000000000000000000000000 2FA

0000

- 0000000000000000 2FA
- 0000 2FA () 00000000
- 0000000000000000 2FA

0000000000

000 2FA 0000000000

- 000000000000
 - 00000000
 - 0000
 - 000000

- 設定項目
2. 設定項目
 3. 設定項目 2FA

設定項目

設定項目

設定項目

設定項目 (✉) 設定項目 設定項目 login_count = 0

設定項目 (✉)

設定項目

設定項目 Bob Wilson (<bob@example.com>)

設定項目 • 設定項目 • 設定項目 • 設定項目 • 設定項目

[設定項目] [設定項目]

設定項目 “設定項目”

設定項目

設定項目 **Mailjet** 設定項目

設定項目 api_crmCommunicationUserWelcome

設定項目

設定項目 (✎) 設定項目

設定項目

設定項目

設定項目

姓名

性别 ☒ admin ☐ customer_service_agent_1 ☐ customer

[姓名] [手机号]

密码

- 密码长度至少8位
- 包含 - 字母/数字/特殊字符
- 包含 /?<?> 特殊字符

电子邮箱

姓名 手机号 () 电子邮箱

姓名

姓名 John Smith (<john@example.com>)

⚠ 密码强度提示

密码强度 • 密码长度 • 2FA 开启 • 密码

密码强度提示

[姓名] [手机号]

姓名 “姓名” 姓名

姓名

姓名

姓名

姓名

- 密码强度提示

- 密碼長度至少 8 個字元
- 密碼中至少包含 3 種字元
- 密碼 2FA 驗證碼

2FA 驗證

- 密碼 2FA 驗證碼
- 密碼長度至少 8 個字元
- 密碼中至少包含 3 種字元
- 密碼 2FA 驗證碼
- 密碼長度至少 8 個字元

密碼

- 密碼長度至少 8 個字元
- 密碼中至少包含 3 種字元
- 密碼 2FA 驗證碼
- 密碼 2FA 驗證碼

密碼

密碼

- 密碼 2FA 驗證碼
- 密碼長度至少 8 個字元
- 密碼中至少包含 3 種字元
- 密碼 2FA 驗證碼
- 密碼 2FA 驗證碼

密碼

- 密碼長度至少 8 個字元
- 密碼中至少包含 3 種字元
- 密碼 2FA 驗證碼
- 密碼 2FA 驗證碼 - 密碼 2FA 驗證碼

2FA 驗證

- 2FA 認證
- 認證
- 認證
- 認證 2FA
- 認證 2FA

認證

- 認證
- 認證
- 認證
- 認證
- 認證

認證

“認證”

- 認證
- 認證
 - 認證
 - 認證
 - 認證
 - 認證

2FA 認證

- 認證
- 認證
 - 認證 → 認證 → 認證
 - 認證 30 分鐘
 - 認證?? 認證
 - 認證
 - 認證 2FA

“Cookie”

- Cookie Cookies
- - Cookies
 - Cookies
 - OmniCRM
 -

-
- -
 -
 - 5-10
 - Mailjet
 -

- 1
- -
 -
 -

2FA

-
- -
 -
 -

##

- 如何 選擇/設置密碼
- 如何設置密碼
 - 密碼長度
 - 密碼複雜度 2FA
 - 密碼有效期間
 - 密碼 2FA 密碼有效期間

如何“設置 2FA”

- 如何 設置
- 如何設置
 - 密碼有效期間
 - 如何 API 設置
 - 密碼有效期間

如何設置密碼

- 如何 API 設置密碼
- 如何設置
 - 密碼有效期間
 - 密碼有效期間
 - 如何 API 設置密碼
 - 密碼有效期間

如何設置

如何設置

- 密碼有效期間
- “密碼”密碼有效期間 30 天
- 密碼有效期間 HTTP-only cookies 密碼有效期間 JavaScript 密碼
- 密碼有效期間 cookies 密碼有效期間 HTTPS 密碼

如何設置

- 密碼有效期間

- 環境構築
- 認証サーバーのインストール
- 認証サーバーの起動

2FA 設定

- TOTP 認証の実装
- 認証サーバーのインストール
- 認証サーバーの起動
- 認証サーバーのテスト

セキュリティ

- 2FA 認証の実装
- 認証サーバーのインストール
- 認証サーバーの起動  
- 認証サーバー IP 設定

デモ

- `2fa` - 2FA API の実装
- `rbac` - RBAC の実装
- `administration_configuration` - Mailjet の設定
- `integrations_mailjet` - Mailjet の連携
- `customer_care` - カスタマーケアの実装



0fT52ZvoZBE

OmniCRM

OmniCRM is a cloud-based customer relationship management (CRM) system.

OmniCRM is a cloud-based customer relationship management (CRM) system.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

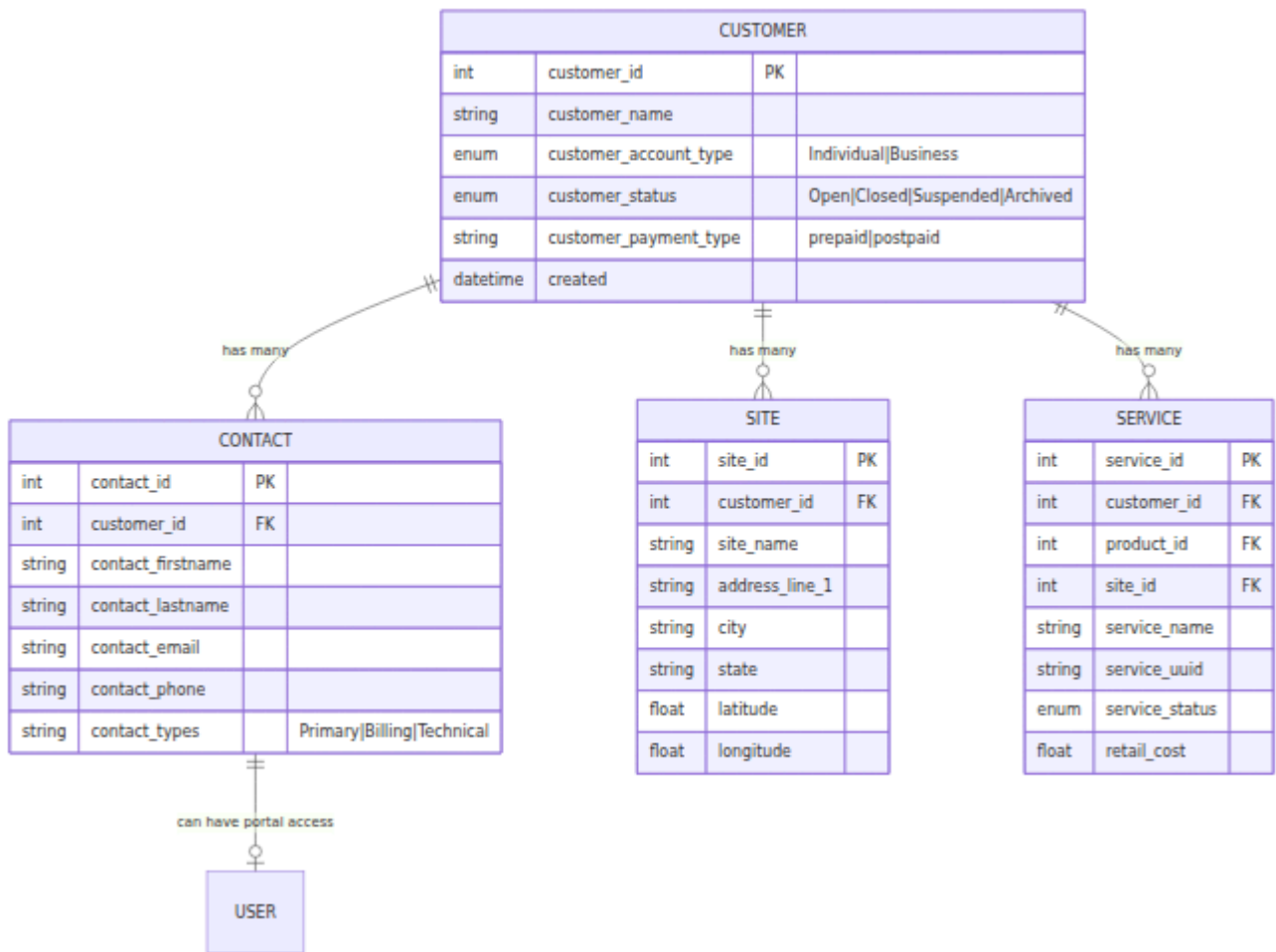
OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

OmniCRM is a cloud-based customer relationship management (CRM) system. It provides a comprehensive view of your customers and their interactions with your business. It also provides tools for managing your sales pipeline, marketing campaigns, and customer support.

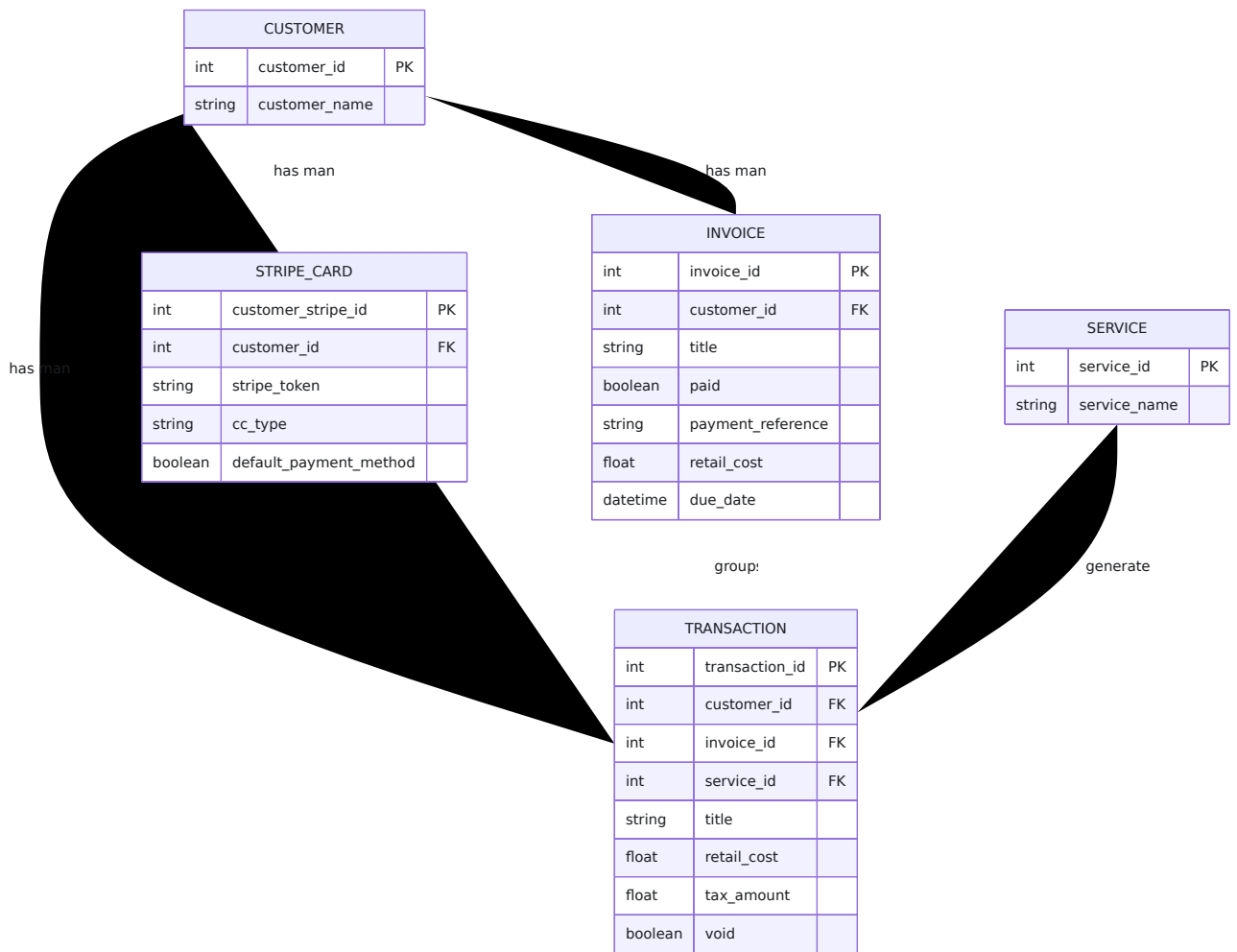


□□□□

- □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□

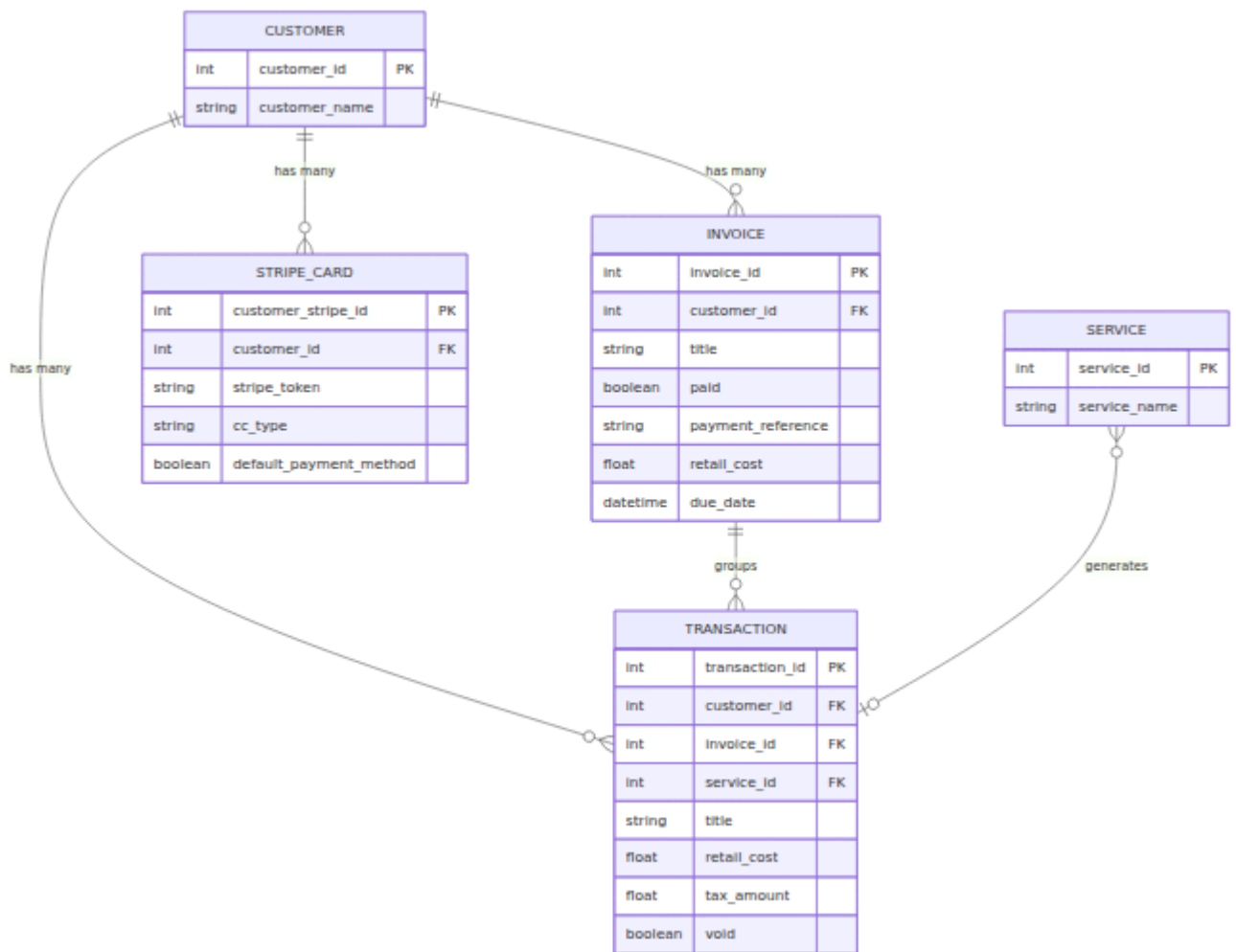


□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- Stripe□□□□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□



□□□□

- □□□□□□□□□□□□□□
- □□□□Ansible□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□□□□□

INVENTORY_TEMPLATE			
int	inventory_template_id	PK	
string	item		SIM Card Router Phone Number
string	itemtext1_label		Serial Number
string	itemtext2_label		MAC Address

CUSTOMER		
int	customer_id	PK
string	customer_name	

SERVICE		
int	service_id	PK
string	service_name	

defines structure

owns

uses

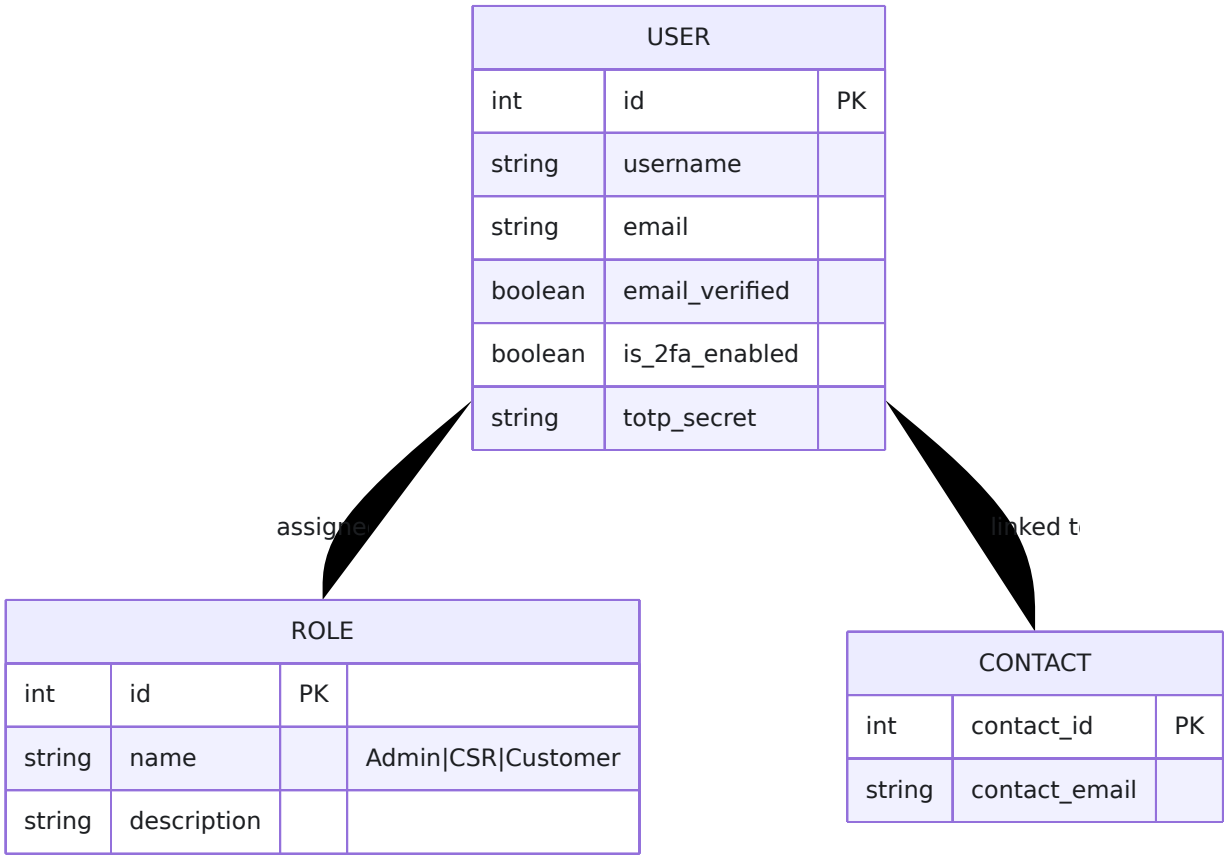
INVENTORY			
int	inventory_id	PK	
int	customer_id	FK	
int	service_id	FK	
int	inventory_template_id	FK	
string	item		
enum	item_state		New Assigned Used Damaged
string	itemtext1		
string	itemtext2		
string	management_url		
text	config_content		

□□□□

- □□□□□□□□□□□□□□□□□□
- □□□itemtext1-20□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□□□



grant:

PERMISSION			
int	id	PK	
string	name		customer.read service.create
string	description		

需求

- 系统支持多用户同时在线
- 系统支持用户2FA验证
- 系统支持用户角色管理
- 系统支持用户权限管理

系统

系统支持多用户同时在线

111

- 00 - 00000ID00000
- 0000 - 00000000000000
- 00 - 0000000000
- 0000 - 0000000000000000

$$\begin{array}{|c|c|} \hline & \\ \hline \end{array} - \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array}$$

□ □

□ □

ARPU

[illegible][illegible]

$$\begin{array}{|c|c|} \hline & \\ \hline \end{array} - \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array}$$

□ □

□ □

[illegible]

Google Maps

[illegible]

□ □ □ □ □ □ □ □ □ □

□ □

1.

- 〇〇〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇Google Maps〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇
 - 〇〇〇〇 - 〇〇Google Maps〇〇〇〇〇
 - 〇〇〇〇**1** - 〇〇〇〇〇〇〇〇〇〇

- **API 2** - 開發者/用戶
- 用戶 - 用戶
- 用戶/用戶 - 用戶
- 用戶 - 用戶
- 用戶 - 用戶
- 用戶 - 用戶
- **Plus** - 11 用戶 8C3MFJV8+2F
- **Google Maps** ID - 用戶

2. 用戶

- 用戶
- 用戶
- 用戶
- 用戶

Plus

Plus 用戶 11 用戶

用戶

用戶 ✓ 用戶 “123 Main Street” ✓ 用戶 1 “123 Main Street” ✓ 用戶 “London” ✓ 用戶 “Greater London” ✓ 用戶 “United Kingdom” ✓ 用戶 “SW1A 1AA” ✓ 用戶 51.5074 ✓ 用戶 -0.1278 ✓ Plus 用戶 “9C3XGPHC+3Q” ✓ Google 用戶 ID “ChIjdd4hrwug2EcRmSrV3Vo6III”

用戶

用戶

- 用戶
- 用戶 REACT_APP_DEFAULT_COUNTRY
- **Plus** 用戶 12 用戶 11 + 1 用戶

用戶

用戶

用戶

1

--	--	--	--	--	--	--	--	--

☐ Plus ☒ OmniCRM

1. □□□□□□

- 000000 - 00Plus0000000000000000
- 0000 - 00000000000000
- 0000 - 000000000000000000

2. □□□□

- 000000 - 0000000000000000/00000000
- 0000 - 0000000000000000 Mailjet <integrations_mailjet>
- 0000 - 000000000000

000

```
SELECT * FROM Customer_Site
WHERE distance(latitude, longitude, 51.5074, -0.1278) < 5
```

□□□47□□□□□□ □□□□47□□□□□□□□□□

3.

- □□□□ - □□□□□□□□□□
- □□□□□□ - □□□□□□□
- □□□□ - □□□□□□□□□□

4.

- □□□ - □□□□□□□
- □□□ - □□□□□□□□□
- □□□ - □□□□□□□□□□□□□□□□

繁體中文

歡迎來到本繁體中文版本

1. 安裝說明

- 安裝說明書
- 安裝說明書/圖解
- 安裝說明書Plus版

2. **Plus**說明

- 安裝Plus說明書 8C3MFJV8+2F
- 安裝Google Maps說明書
- 安裝Plus說明書

3. 其他

- “繁體中文”說明書
- “繁體中文說明書500”
- 其他說明書

繁體

歡迎來到本繁體中文版本

OmniCRM

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

[illegible]



WebApp Web

□ □

□ □



--	--	--	--

[illegible]

□ □

[illegible][illegible]

```
0000000000 00 <payments_invoices>0000000000
```

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □

1. 0000 0000 → [00000000]
2. 00 00 00
3. 0000000000000000

□□□□□□

1. □□□ □□ → □□□□□□□□
2. □□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□□□□

□□□□□□

- □□□□ - □□□□□□□□□□□□□□□□
- □□□□□□ - □□□□□□□□□□□□
- □□□□□□ - □□□□□□□□□□□□
- □□□□□□ - □□□□□□□□□□□□

□□□□□□

- □□1,000□□□□□□“k”□□□□□□\$1.5k□
- □□1,000,000□□□□□□“M”□□□□□□\$2.3M□
- □□1,000,000,000□□□□□□“B”□□□□□□\$1.1B□

□□□□

□□□□□□□□□□□□□□

□□□□

- **ID** - □□□□ID
- □□ - □□□□□□
- □□ - □□□□□□
- □□ - □□□□□□□□
- □□ - □□□□□□□□□□□□□□
- □□ - □□□□□□□□□□□□ID□□□□□□
- □□ - □□□□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□ : □□□□□□□

- □□□□ - □□□□□□□□□□
- □□□□**PDF** - □□PDF□□□□□□□□
- □□□□ - □□□□□□□□□□□□□□□□

□□□□

□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□

- □□□□□□ - □□□□□□□□□□□□
- □□□□□□ - □□□□□□□□
- □□□□ - □□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□SIM□□□□
- □□□□□□ - □□□□□□□□

- 0000 - 0000000000

00000000/0000

00000000000000

- 0000 - 00000000
- 000 - 0000000/000000
- 0000000 - 000000000000
- 0000 - 00000000
- 00 - 00000000
- 00 - 0000000000

00000000

00 "+ 0000" 000000000000

0000000000

00000000/0000

အခြေခံ

- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ

အခြေခံ

- အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံအားဖြင့်/အခြေခံ

အခြေခံ

1. အခြေခံအားဖြင့်/အခြေခံ
2. အခြေခံအားဖြင့်/အခြေခံ

3. 請將“ ”填上
4. 請將填上
5. 請將填上

請將填上

請將

請將填上

- ID
-
-
- ID

請將

請將填上

請將填上

- -
- -

請將填上

- -
- -

請將

請將填上

- **ID** - ID/
- -
- -

- 00 - 000000000000/0000
- 00 - 0000ID00

0000000000000000 ↔ 00000

00000

000000000000 000000000000

0000

- 00000000
- 00000000000000000000

000000

1. 0000000000
2. 00000000 : 0
3. 00 "0000"
4. 00000000

0000000

- 000000 void = true
- 00000000000000

- 零售店
- 零售店“店”名称
- “零售店”名称

零售店 零售店名称“零售店名称”零售店“零售店名称”

零售店

零售店零售店零售店零售店零售店零售店零售店零售店零售店零售店

零售店

- 零售店 - 零售店
 - 零售店 - 零售店零售店零售店零售店零售店
 - 零售店 - 零售店零售店零售店零售店
 - 零售店 - 零售店 $\text{retail_cost} \times (\text{tax_percentage} / 100)$
 - 零售店 - 零售店零售店零售店 \$10.00 (10%)
- 零售店/零售店 - 零售店
 - 零售店零售店零售店
 - 零售店零售店零售店零售店0%
 - 零售店零售店零售店零售店零售店零售店

零售店

- 10% of \$50.00
- $\$50.00 \times 0.10 = \5.00
- \$5.00 (10%)

NIL

- 0
- 0%
- “-”

-
-
-
- “”

- 00000000-0

000000

- 000000
- 0000000000000000
- 00ID000000000000
- 00000000✓0
- 0000

00000

000000000000

1. 0000000000000000
2. 000000000000
3. 000000000000
4. 00 invoice_id 00000
5. 00000000“0000”

0000 payments_invoices 0000000000000000

00000000

00000**1**00000000000000

1. 000000“00000020”
2. 0000000£10000
3. 000000 00 00
4. 00 "+ 00000"
5. 00 00 0000
6. 00 0000 0000
7. 000000“00000000”
8. 000000“0108-90000000”
9. 0000000010.00

10. 0000000000000000
11. 00 "0000"
12. 000-£10.00000000
13. 0000000000000000

0000**2**0000000000

1. 000000000000
2. 0000000£75000000
3. 000000 00 00
4. 00 "+ 0000"
5. 00 00 0000
6. 000000"0000"
7. 000000"00000000000000"
8. 0000000075.00
9. 0000000045.00000000000000
10. 0000000000
11. 00000000
12. 00 "0000❖❖❖"
13. 00000000000000
14. 000000000000

0000**3**0000000000

1. 000000000000
2. 000000000000
3. 000000000000000 : 0
4. 00 "0000"
5. 0000000000
6. 000000000000
7. 000000000000

□□□□

- `payments_invoices` - □□□□□□
- `payments_process` - □□□□□□
- `basics_payment` - □□□□□□
- `csa_activity_log` - □□□□□□□□□□

--	--	--	--

```
</payments_transaction> 
```

Mailjet <integrations_mailjet> PDF

環境構築

OmniCRMのHTMLテンプレートはJinja2テンプレートエンジンを使用しています。

テンプレート

テンプレートは `OmniCRM-API/invoice_templates/`

にあります。

- `norfone_invoice_template.html` - ノーフォンのテンプレート
- `cifi_invoice_template.html` - CIFのテンプレート

また、

テンプレートは `OmniCRM-API/crm_config.yaml` に定義されています。

```
invoice:
  template_filename: 'norfone_invoice_template.html'
```

変数

テンプレートにはJinja2変数があります。

変数は

- `{{ invoice_number }}` - 請求ID `INV-2025-001234`
- `{{ date }}` - 発行日 `2025-01-10T12:00:00`
- `{{ due_date }}` - 支払期限 `2025-02-10`
- `{{ start_date }}` - 有効開始日 
- `{{ end_date }}` - 有効終了日
- `{{ total_amount }}` - 総額
- `{{ total_tax }}` - 総税額

また、

- `{{ client.name }}` - 顧客名

- `{{ client.address.address_line_1 }}` - 0001
- `{{ client.address.address_line_2 }}` - 0002
- `{{ client.address.city }}` - 00
- `{{ client.address.state }}` - 0/0
- `{{ client.address.zip_code }}` - 0000
- `{{ client.address.country }}` - 00

000000

0000000000000000

```
{% for sub_transaction in transactions %}
  <tr>
    <td>{{ sub_transaction.transaction_id }}</td>
    <td>{{ sub_transaction.created.split("T")[0] }}</td>
    <td>{{ sub_transaction.title }}</td>
    <td>{{ sub_transaction.description }}</td>
    <td>${{ "%.2f"|format(sub_transaction.retail_cost) }}</td>
  </tr>
{% endfor %}
```

000000

- `sub_transaction.transaction_id` - 00ID
- `sub_transaction.created` - 0000/00
- `sub_transaction.title` - 0000
- `sub_transaction.description` - 0000
- `sub_transaction.retail_cost` - 000000
- `sub_transaction.tax_percentage` - 0000000000100010%0
- `sub_transaction.tax_amount` - 0000000000

0000000000

```

<td>
    {% if sub_transaction.tax_amount and
sub_transaction.tax_amount > 0 %}
        ${{ "%.2f"|format(sub_transaction.tax_amount) }} ({{
sub_transaction.tax_percentage }}%)
    {% else %}
        -
    {% endif %}
</td>

```

□□□□□□□□

□□**1**□□□□□□

```

cd OmniCRM-API/invoice_templates/
cp norfone_invoice_template.html
your_company_invoice_template.html

```

□□**2**□□□□**HTML/CSS**

□□ `your_company_invoice_template.html` □□□□□□□□

□□□□□□□□

1. □□□□□□□□

```

<!-- □□□□□URL□□ -->
![[□□□□]](https://yourcompany.com/logo.png)

<!-- □□□□□□□□ -->
<h1>□□□□□□□□</h1>

```

2. □□□□

```

<style>
  /* Navbar */
  .navbar {
    background: linear-gradient(to bottom right, #your-color-1, #your-color-2);
  }

  /* Table */
  .table thead th {
    background-color: #your-brand-color !important;
    color: white !important;
  }

  /* Button */
  .btn-primary {
    background-color: #your-brand-color;
  }
</style>

```

3. Footer

```

<footer>
  <p>Footer Content</p>
  <p>123 Street, City, State, Zip</p>
  <p>Phone: +1-555-123-4567 | Email: billing@yourcompany.com</p>
  <p>ABN/12345678900</p>
</footer>

```

4. Links

```
<div class="payment-info">
  <h3>결제정보</h3>
  <p><strong>결제 URL</strong> | https://yourcompany.com/pay |</p>
  <p><strong>계좌입금 정보</strong></p>
  <ul>
    <li>계좌번호: 1234567890123456</li>
    <li>BSB: 123-456</li>
    <li>계좌번호: 987654321</li>
    <li>계좌번호: {{ invoice_number }}</li>
  </ul>
</div>
```

5. 결제조건

```
<div class="terms">
  <h4>결제조건</h4>
  <p>결제기한: 30일 이내</p>
  <p>결제수수료: 2%</p>
  <p>결제문의: billing@yourcompany.com</p>
</div>
```

3. 결제조건

OmniCRM-API/crm_config.yaml

```
invoice:
  template_filename: 'your_company_invoice_template.html'
```

4. API

```
cd OmniCRM-API
sudo systemctl restart omnicrm-api
```

5. 테스트

1. 결제조건 테스트
2. 결제조건 테스트

3. PDF

4.

Jinja2

```
{% if total_amount > 1000 %}
    <div class="high-value-notice">
        <p><strong></strong> - </p>
    </div>
{% endif %}

{% if client.address.country == "Australia" %}
    <p><strong>GST ${{ "%.2f"|format(total_amount * 0.10) }}</p>
{% endif %}
```

```
invoice_template_en.html
invoice_template_es.html
invoice_template_fr.html
```

```

<!-- 结账金额 -->
<tr>
    <td colspan="4" class="text-right"><strong>总计</strong></td>
    <td>${{ "%.2f"|format(total_amount) }}</td>
</tr>
<tr>
    <td colspan="4" class="text-right"><strong>税金</strong></td>
    <td>${{ "%.2f"|format(total_tax) }}</td>
</tr>
<tr>
    <td colspan="4" class="text-right"><strong>总计</strong></td>
    <td>${{ "%.2f"|format(total_amount + total_tax) }}</td>
</tr>

```

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并使用了 `tax_percentage` 模板变量，并使用了 `0%` 模板变量。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并使用了 `tax_percentage` 模板变量，并使用了 `0%` 模板变量。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并使用了 `tax_percentage` 模板变量，并使用了 `0%` 模板变量。

```

<div class="qr-payment">
    ![[二维码]](https://api.qrserver.com/v1/create-qr-code/?
    size=150x150&data={{ payment_url }})
    <p>二维码</p>
</div>

```

PDF 模板

OmniCRM 使用 **WeasyPrint** 将 HTML 模板转换为 PDF 文档。

在 CSS 中，

- 使用 CSS 2.1 规范
- 使用 CSS3 的 flexbox 布局
- 使用 `@font-face` 定义字体

在 HTML 中，

- JavaScript
- CSS Grid
-
- CSS

```
@page {  
  size: A4;  
  margin: 1cm;  
}  
  
body {  
  font-family: Arial, sans-serif;  
  font-size: 10pt;  
}
```

```
@media print {  
  .no-print {  
    display: none;  
  }  
  
  .page-break {  
    page-break-after: always;  
  }  
}
```

```
.table {
  table-layout: fixed;
  width: 100%;
}

.table th, .table td {
  word-wrap: break-word;
  padding: 4px;
}
```

□□□□□

□□□□□□□◆◆□□□□□□□□□□□□

```
@font-face {
  font-family: 'YourFont';
  src: url('https://yourcompany.com/fonts/yourfont.woff2')
  format('woff2');
}

body {
  font-family: 'YourFont', Arial, sans-serif;
}
```

□□□□□□□

□□□□□

1. □□□□□
 - □□□□□□
 - □□□□□□□□
 - □□□□□□□□□□
 - □□□□□□□
 - □□□□□□□
2. □□□□□□□
 - □□□□□□□□□□
 - □□□□□□□□□□

- ### 3. PDF

4.

5.

- [illegible]

□□□□API□□□□□□□□

□ □ □ □ □

□ □ □ □ □ □

- PDF 00000

- 2004 2005 CSS 2.1
- 2004 2005 CSS 2.1 WeasyPrint CSS

2. 查詢 - 查詢所有發票
 - 查詢invoice_id查詢PDF
 - 查詢所有發票查詢PDF
 - 查詢 `last_accessed` 查詢所有發票
3. 查詢 - 查詢所有發票PDF查詢
 - 查詢所有發票/查詢所有發票
 - 查詢所有發票
 - 查詢所有發票

查詢

- 查詢 - 查詢所有發票PDF查詢所有發票
- 查詢 - 查詢所有發票PDF查詢所有發票
- 查詢 - 查詢PDF查詢CPU查詢
- 查詢 - 查詢所有發票查詢所有發票

查詢

查詢PDF查詢所有發票查詢所有發票查詢所有發票查詢所有發票

- 查詢所有發票90查詢所有發票
- 查詢所有發票30查詢所有發票
- 查詢所有發票查詢所有發票

API查詢

查詢API查詢UI查詢所有發票

- 查詢所有PDF查詢所有發票查詢所有發票
- 查詢所有發票查詢所有發票
- 查詢所有發票查詢所有發票

查詢 查詢所有發票查詢所有發票查詢所有發票查詢所有發票

```
-- 查詢所有發票PDF查詢MySQL查詢  
DELETE FROM Invoice_PDF_Cache;
```

📄 `crm_config.yaml` 📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

1. 📄📄 📄 → **📄📄📄**
2. 📄 📄 📄
3. 📄📄📄📄📄📄📄📄

📄📄📄📄

1. 📄📄 📄 → 📄📄📄📄📄
2. 📄📄📄📄📄📄

📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

- 📄📄📄 - 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

- 00000 - 00000000000000000000
- 00000 - 0000000000000000
- 00000 - 0000000000000000

□ □ □ □ □

- 1,000 "k" \$1.5k
- 1,000,000 "M" \$2.3M
- 1,000,000,000 "B" \$1.1B

□ □ □ □ □ □

- “ ” “ ”
-
-
-

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□

- **ID** - 00000ID
- 00 - 0000/00
- 00 - 0000000000 - 0000000"000"000"N/A"
- 000 - 000000
- 00 - 0000000
- 00 - 0000000000000
- 00 - 0000000000000
- 00 - 00000000000000

000000

- ↓ (00) - 00000PDF
- 0 (00) - 0000000000000
- 0 (00) - 000000000000000
- ☒ (0000) - 0000000000000
- 0 (00) - 00Stripe000000000000Stripe000

000000

00 "+ 0000000" 0000000

000000

--	--	--	--	--	--

□ □

□□□□□□□□

- 0000 - 0000ID0000000000000000
- 0000 - 000000000000000000
 - 0000
 - 00000
 - 0000
 - 0000000000 \$10.00 (10%)
 - 0000000000“-”
- 0000 - 00000000
 - 0000
 - 00000000000000
 - 0000000000
 - 00000000 + 00
- 0000 - 000000000000
 - 00**PDF** - 0000PDF00000000
 - 00000000 - 000000000000000000000000
 - 0000 - 00000000000000000000
 - 00 - 00Stripe000000000000Stripe0000

- PDF - 文件格式

PDF

PDF 文件格式 ↓ 文件扩展名为 ".PDF" 的文件格式

PDF 文件

- 文件名称
- 文件扩展名
- 文件内容 (Invoice_01234.pdf)
- PDF 文件

PDF 文件

- 文件 - PDF 文件 (2-3 页)
- 文件 - 文件
- 文件 - 文件

PDF 文件

- 文件 - API 文件
- PDF 文件 - 文件
- 文件 - 文件

PDF

PDF 文件

목차

1. 결제시스템의 종류
2. 결제  결제시스템의 종류
3. 결제시스템
 - **Stripe** - 결제시스템의 종류
 - **Paycom** - 결제시스템의 종류
 - **Paycom** - 결제시스템의 종류
 - **POS** - 결제시스템의 종류
 - **결제** - 결제시스템의 종류
4. 결제시스템 Stripe
 - 결제시스템의 종류
 - 결제시스템
 - 결제시스템
5. 결제시스템의 종류
 - 결제시스템의 종류
6. 결제 "결제" 결제
7. 결제시스템
 - **Stripe** - 결제 Stripe API 결제
 - 결제 - 결제시스템의 종류
8. 결제시스템 "결제"
9. 결제시스템

2. 如何將PDF文件轉為可編輯格式

```
3. Mailjet <integrations_mailjet> api_crmCommunicationCustomerInvoice
```

4.

- PDFファイル
- 画像
- テキストファイル
- 音声ファイル
- 動画ファイル
- 圧縮ファイル

5. □□□□“□□□□□□□□□□”

□ □ □ □ □ □ □ □

[illegible]

□□□□□□□□

- `{{ var:customer_name }}` - 客户名称
- `{{ var:invoice_number }}` - 发票ID
- `{{ var:invoice_date }}` - 开票日期
- `{{ var:due_date }}` - 到期日期
- `{{ var:total_amount }}` - 总金额
- `{{ var:invoice_url }}` - 发票/回单PDF链接
- `{{ var:pay_url }}` - 付款二维码

□ □ □ □ □ □ □ □ □ □

- **環境変数** - `crm_config.yml` Mailjet API
- **環境変数** - `crm_config.yml`
- **PDF** - `PDF`

4444

□ □ □ □ □ □ □ □ □ □ □ □ □

- 関数 = “Stripe API” を呼び出す

Stripe

Stripe を呼び出す

関数

- 関数の名前
- 関数の名前 **Stripe**
- 関数の引数 `payment_reference` Stripe の ID
- 関数の戻り値

関数

1. Stripe を呼び出す
2. Stripe の ID を取得
3. Stripe の ID を取得

4. 関数 “Stripe”

5. Stripe の ID を取得
 - Stripe API を呼び出す
 - Stripe の ID を取得
 - 関数の引数 `refund_reference`

6. 関数の戻り値 “Stripe”

7. 準備作業

準備作業

- 準備作業の準備作業
- 準備作業“準備”
- 準備作業
- 準備作業の準備作業3-7準備作業
- Stripe準備作業

準備作業

- 準備作業のPOS準備作業の準備作業
- 準備作業の準備作業
- 準備作業

準備作業

準備

準備作業の準備作業

- ID
- 準備
- 準備作業の準備作業

準備

準備作業の準備作業

準備作業

- 準備 - 準備作業
- 準備 - 準備作業

準備

7. 000000

- 0000000000
- 0000000000000000
- 000000000000

8. 00 "0000"000000000000"000000150"

9. 0000000000000000

10. 0000000000000000

11. 0000000000 "000000" 00000000000000

12. 000000PDF00000000000000

00002000000000

1. 00000000000+00000000000

2. 0000000000000000

3. 0000000000000000

- 00 "+ 000000"
- 000"00000 - 2025010"
- 00/00010100310
- 0000020150
- 000000000000000000
 - 000000000 X 00
 - 000000000 X 00
 - 0000000000
- 000000000000
- 00 "0000"000000000000

4. 0000000000000000

- 0000 "+ 000000"
- 000"00000 - 2025010"
- 00/000101003100000000
- 00000000
 - 0000000000000000
 - 00 X 000000000
 - 0000000000000000
- 00 "0000"

5. 0000000000000000
 - 0000 "+ 0000000"
 - 000"00000 - 202501"
 - 0000000000000000
 - 00 "00" 00000
 - 00 "0000"
6. 00000000000000000000

0000**3**000000000000

1. 000000
2. 00000 00 00
3. 00 "+ 0000000"
4. 000000000
5. 000000020000
6. 00000000000000000000
7. 00000000
 - 00000000000"0000000"
 - 00 **X** 0000000
 - 000000000000"00000"
 - 00 **X** 0000000
 - 00000000"000018000"
 - 000000000
8. 000000000000000000
9. 00 "000000**18**"
10. 000000000000000
11. 00/0000000000000000000000

0000**4**000000000000

1. 000000000
2. 00000000000000
3. 000000000000
 - 00000000000000

- 𐀀 **X** 𐀀𐀀𐀀𐀀
- 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
- 𐀀 **X** 𐀀𐀀𐀀𐀀𐀀

4. 𐀀𐀀𐀀𐀀𐀀
5. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
6. 𐀀 "𐀀𐀀𐀀"𐀀𐀀𐀀𐀀𐀀𐀀𐀀
7. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀/𐀀𐀀𐀀𐀀𐀀
8. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀

𐀀𐀀𐀀**5**𐀀𐀀𐀀𐀀𐀀𐀀𐀀

1. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
2. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
3. 𐀀𐀀𐀀𐀀 𐀀𐀀 𐀀𐀀
4. 𐀀 "+ 𐀀𐀀𐀀𐀀𐀀"
5. 𐀀𐀀𐀀𐀀𐀀𐀀
 - 𐀀𐀀𐀀𐀀𐀀𐀀
 - 𐀀𐀀𐀀𐀀𐀀𐀀
 - 𐀀𐀀𐀀𐀀𐀀𐀀 + 7𐀀
 - 𐀀𐀀𐀀“𐀀𐀀𐀀𐀀”
6. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
7. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀
8. 𐀀 **X** 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
9. 𐀀 "𐀀𐀀𐀀"
10. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
11. 𐀀𐀀𐀀Stripe𐀀𐀀𐀀

𐀀𐀀𐀀**6**𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀

1. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
2. 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
3. 𐀀𐀀𐀀𐀀 𐀀𐀀𐀀𐀀𐀀𐀀𐀀𐀀
4. 𐀀𐀀𐀀𐀀𐀀𐀀
 - 𐀀ID𐀀𐀀𐀀𐀀𐀀

- 0000000000
- 0000000000
- 0000000000

5. 00000000000000
6. 000000PDF00000000 "00**PDF**" 00
7. 000000000000000000 "000000" 00
8. 0000000000

0000**7**0000000000

1. 0000000000
2. 000000000000
3. 0000000000000000
4. 00000000
5. 00000000000000
6. 000000000 "00" 00
7. 00000
8. 00000000000
9. 000000000000000000
10. 0000000000000000
 - 00000000000000000000
 - 0000000000
11. 00000000000000000000

0000**8**0000000000

1. 00000000000000
2. 000000 00 00
3. 0000000000
4. 0000000000000000
5. 0000000000
6. 0000000000 "00000" 00
7. 00 "00" 00000
8. 000000"000002025-01-15"

Stripe

- Stripe のインストール
- Stripe の API キーの取得と Stripe のテストモードの起動

Mailjet

- Mailjet API のインストールとテストモードの起動
- `crm_config.yaml` に Mailjet の API キーを設定する

Stripe

- Stripe の API キーの取得と Stripe のテストモードの起動
- Stripe の API キーの取得と Stripe のテストモードの起動

PDF

- PDF の生成と保存
- PDF の削除 `DELETE FROM Invoice_PDF_Cache WHERE invoice_id = X;`

Stripe

- Stripe の API キーの取得と Stripe のテストモードの起動
- Stripe の API キーの取得と Stripe のテストモードの起動

Mailjet

- Mailjet API のインストールとテストモードの起動
- Mailjet の API キーの取得と Mailjet のテストモードの起動

Stripe

- Stripe の API キーの取得と Stripe のテストモードの起動
- Stripe の API キーの取得と Stripe のテストモードの起動

Mailjet

- Mailjet API のインストールとテストモードの起動
- Mailjet の API キーの取得と Mailjet のテストモードの起動

○○○○○○

OmniCRM○○ StripePayPal○○ OmniCRM○○

○○○○○○ <payment_system_guide>○○○○○○ <billing_overview>○○○○○○ <payments_process>○○○○ <payments_invoices>○○

○○

○○○○○○○○○○

- ○○○○○○ - ○○○○○○○ StripePayPal○○○○○○○○○○○○○○○○○○ OmniCRM
- ○○○○○○ - ○○ StripePayPal○○○○
- ○○○○ - ○○○○○○○○○○○○○○○
- ○○○○ - ○○○○○○○○○○○○○○○○○○○
- ○○○○ - ○○○○○○○○○○○○○○○
- ○○○○ - ○○○○○○○○ <self_care_portal> ○○○○○○
- ○○○○ - ○○○○○○○○○○○○○○○/○○○

○○○○○○○○○○

- ○○○○ VisaMastercardAmerican ExpressDiscover
- ○○○○
- ○○○○○○○○○○○○○○○

○○ **OmniCRM** ○○○○

○○ OmniCRM○○○○○○

- ○○○○○○ stripepaypal
- ○○○○ VisaMastercard○○
- ○4○○○
- ○○○/○
- ○○○○○/○○
- ○○○○○○○○○○○○○○○

□□□□□□

□□□□□□

1. □□□ □□ → **【□□□□□】**
2. □□ □□ □□
3. □□□ □□□□ □□

□□□□

□□□□□□□□□□

□□□□□□□□□□□□

□□□□□□□□□□□□60□□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□

□□□□

- □□ - □□□□□□□□□□“□□□”□“□□Visa”□
- □□□ - □□□□□4□□□
- □□ - □□□/□□MM/YYYY□□□

- [API - 如何集成支付](#)
- [API - 如何集成支付](#)

如何集成支付

如何集成支付 : 如何集成支付

- [如何集成支付 - 如何集成支付](#)
- [如何集成支付 - 如何集成支付](#)

如何集成支付

如何 "如何集成支付" 如何集成支付

如何**1**如何集成支付

如何集成支付Stripe ElementsPayPal SDK如何集成支付

如何集成支付

- [如何集成支付 - 如何集成支付CVC如何集成支付Stripe如何集成支付](#)
- [如何集成支付 - 如何集成支付](#)
- [如何集成支付 - 如何集成支付](#)

項目

- 項目 - 項目項目項目項目

項目

- 項目項目項目項目項目項目iframe項目Stripe Elements / PayPal SDK
- OmniCRM項目項目項目項目項目
- PCI DSS項目項目項目項目項目
- 項目項目項目項目

項目2項目項目項目

項目 "項目項目" 項目

- 項目項目
 - 項目項目項目項目
 - 項目項目項目項目
 - 項目CVC項目
- 項目項目
 - 項目項目項目項目項目項目項目項目OmniCRM
 - 項目項目項目項目項目項目pm_1A2B3C4D項目Stripe
 - 項目項目OmniCRM
- 項目項目
 - OmniCRM項目項目項目項目項目項目項目項目
 - 項目4項目項目項目項目項目項目項目項目
 - 項目項目項目項目項目項目OmniCRM項目

項目3項目項目

項目項目項目

項目Visa項目項目1234項目項目項目項目

項目項目項目項目項目

項目項目項目

- 國際標準化組織
- 國際標準化組織
- 國際標準化組織

國際標準化組織

國際標準化組織

- 國際標準化組織
- 國際標準化組織
- 國際標準化組織
- 國際標準化組織

國際標準化組織

1. 國際標準化組織
2. 國際標準化組織 : 國際標準化組織
3. 國際標準化組織
4. 國際標準化組織

Visa國際標準化組織5678國際標準化組織

國際標準化組織

國際標準化組織

國際標準化組織

國際標準化組織

國際標準化組織

第1部分

- 项目背景与目标
- 项目范围与边界
- 项目组织与沟通

第2部分

项目概述

项目背景与目标

项目范围与边界

项目组织与沟通

项目背景与目标

项目范围与边界

第3部分

项目概述

项目背景与目标

项目范围与边界

- 项目背景与目标
- 项目范围与边界
- 项目组织与沟通

项目总结

OmniCRM项目总结

XXXXXXXXXX

XXXX XX → XXXXXXX XXXXXXXXXXXX

XX XXX XXXXXXXXXXX John Smith Visa **1234 02/2025 12 XX Acme Corp
MC5678 03/2025 45 XX Jane Doe Amex**9012 01/2025 XXX XX

XXXX

- XXXX - XXXX30/60/90XXXX
- XXXX - XXXX
- XXXX - XXXXXXXXXXXXXXX

XXX

- XX - XXXXXXXXXXXXXXX
- XX - XXXXXXXXXXXXXXXMailjet

XXXX

XXXXMailjetXXXXXXXXXX

- XXX60X - XXXX

- 30日 - 30日
- 7日 - 7日
- 1日 - 1日

30日 - 30日

7日 - 7日

Mailjet

- 30日
- 7日
- 1日
- 1日

integrations_mailjet

30日 - 30日

7日 - 7日

1. 30日 - 30日
2. 7日 - 7日
3. 1日 - 1日
4. 1日 - 1日
5. 1日 - 1日
6. 1日 - 1日
7. 1日 - 1日

30日 - 30日

7日 - 7日

1. 30日 - 30日
2. 7日 - 7日
3. 1日 - 1日
4. 1日 - 1日

5. □□□□□

6. □□□□□

11

[illegible]

- 
- 
- 

□ □ □ □ □ □ □ □ □

□□□□□□□□□□□□□□□□

□□□□□

1.

- ☐ 
- ☐ 
- ☐ 
- ☐ 

2. □□□□

- ☐ □□□□□
- ☐ □□□□□□□
- ☐ □□“□□□□□”□□□□□□□□□□□□□□□□

3.

- 000000 - 0000000000000000
- 000000 - 000000000000
- 00000000 - 00000000000000

555

1-3

- 
- 

- □□□□□□□□□□

4-7

- 
- 
- 

8-14

- 
- 
- 

15

- 10000000000000000
- 1000SIM10000000000000000
- 10000000
- 10000000000000000

□□□□□□□

□□□□□□□□

- 30
-
-
-

□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □

1.
2.
3.
4.

5. 銀行振込

銀行振込

銀行振込

OmniCRM銀行振込

1. 銀行振込 → 銀行振込
2. 銀行振込 → 銀行振込
3. 銀行振込OmniCRM → 銀行振込
4. 銀行振込 → 銀行振込

OmniCRM銀行振込

```
{
  "vendor": "stripe",
  "vendor_payment_method_id": "pm_1A2B3C4D5E6F",
  "payment_type": "card",
  "brand": "visa",
  "last4": "1234",
  "exp_month": 12,
  "exp_year": 2026,
  "name": "John Smith",
  "nickname": "銀行",
  "is_default": true
}
```

OmniCRM銀行振込

- 銀行振込
- CVV/CVC
- 銀行振込
- PIN

PCI

- **PCI** - OmniCRM
- - iframe
- -
- - HTTPS/TLS

<payment_system_guide>

1

- 1.
- 2.
- 3.
4. Stripe
- 5.
- 6.
- 7.
- 8.

2

API 仕様

この API は、お客様のシステムと当社のシステム間でデータをやり取りするためのものです。

API の仕様は、`<payment_methods>`、`<payments_transaction>`、`<payments_invoices>`、`<payments_process>` の 4 つの API から構成されています。

API の概要

API の概要は、以下の通りです。

1. API の名称 → **【API 名称】**
2. API の URL
3. API の説明

API の詳細は、以下の通りです。

API の詳細は、以下の通りです。

- API → API - API の説明
- API → API - API の説明

API の詳細は、以下の通りです。

API `<self_care_portal>` は、お客様のシステムと当社のシステム間でデータをやり取りするためのものです。

- API の説明
- API の説明
- API の説明

API の詳細

API の詳細は、以下の通りです。

1

2

3

- 4
- 5
- 6 Stripe
- 7

📁 basics_payment

📁 2📁

📁 📁📁📁📁📁📁

📁📁📁

- 📁📁📁📁📁📁📁📁📁📁
- 📁📁📁📁📁📁/📁📁📁📁📁
- 📁📁📁📁📁📁📁
- 📁📁📁📁📁
- 📁📁📁📁📁📁📁📁

📁 payments_transaction

📁 3📁

📁 📁📁📁📁📁📁📁

📁📁📁

- 📁📁📁📁📁📁📁📁📁📁
- 📁📁📁📁📁📁/📁📁📁📁📁
- 📁📁📁📁📁📁
- 📁📁 PDF
- 📁📁📁📁📁📁📁📁
- 📁📁📁📁 Stripe 📁📁📁📁📁

- 数据库

数据库 `payments_invoices`

数据库

数据库

数据库

数据库 → 数据库

1. 数据库

数据库

- 数据库
- 数据库
- 数据库 `invoice_id` 数据库 `null`

数据库

2. 準備

必要なもの

- 1 つのアカウント
- 開発環境 (Python 3.6+)
- 環境変数 `invoice_id` を設定
- 必要なライブラリ

完了

3. 実行

- 実行コマンド
- 出力結果
- エラーメッセージ

実行 → 結果

1. 実行

必要なもの

- 実行コマンド `Stripe` を実行
- 実行環境 (Python 3.6+)
- 必要なライブラリ

2. 実行結果

実行コマンド `Stripe`

- 実行結果
- 実行コマンド `#1234` を実行
- 実行結果 `-$45.00` を実行
- 環境変数 `invoice_id` を実行

完了

3. 準備

- 準備するもの
- 準備する環境

準備 → 実行

Stripe 準備

1. Stripe アカウントを作成
2. Stripe 開発者アカウントを作成
3. Stripe 開発者アカウントの API キーを取得
4. Stripe 開発者アカウントの API キーを環境変数に設定
5. Stripe 開発者アカウントの API キーを環境変数に設定
6. `payment_reference` を Stripe の ID

Stripe 準備

1. Stripe/POS/Stripe 開発者アカウント
2. Stripe 開発者アカウント
3. Stripe 開発者アカウント
4. Stripe 開発者アカウント
5. Stripe 開発者アカウント

Stripe 準備

Stripe 準備

Stripe 1 Stripe 開発者アカウント

Stripe 準備

1. Stripe 開発者アカウント
 - Stripe → Stripe 開発者アカウント
 - Stripe 開発者アカウント → Stripe "Stripe 開発者アカウント"
 - Stripe Stripe 開発者アカウント

- 顧客の住所を登録

2. 商品

- 商品名と価格を設定
 - 商品名: \$50.00
 - 商品名: \$45.00
 - 商品名: \$95.00

3. 注文

- 商品名 → 商品 "商品名"
- 商品名と価格を設定
- 商品 "商品名"
- 商品 #INV-2025-001234 商品名 \$95.00

4. 注文

- 商品名と価格を設定 #INV-2025-001234
- 商品名 商品名 \$0.00

5. 商品名

- 商品名と価格を設定
- 商品名 PDF 商品名と価格を設定

6. 商品名

- 商品名と価格を設定
- 商品名と価格を設定
- 商品 "商品名" 商品
- 商品名と価格を設定
- 商品 "商品名"
- Stripe 商品

7. 商品名

- 商品名と価格を設定 "商品名"
- 商品名 商品名 \$95.00

商品名 商品名と価格を設定

商品名 2商品名

商品名 商品名と価格を設定

1. 商品名

- 000000103100
- 00000000000000000000
- 00 000000000000

2. 0000000000

- 000 00 00
- 000 0000000000
- 0000000000000000
- 00000000000000

3. 000000

- 000 00 → 0000000000
- 00   0000000000000000
 - 00 "000000"
 - 0000
 - 0000002025-01-01
 - 0000002025-01-31
 - 0000002025-02-15
 - 00 "0000"

4. 000000

- 00000000000000
- 000000 000000 \$0.00

5. 00000000000000

- 00000000000000000000
- 00000000000000

6. 000000

- 00000000000000000000000000000000
- 00000000000000000000/POS 00
- 000000 0000000000000000

0000 000000100000000000000000000000000000

000000 **3**00000000000000

0000 000000000000000000000000000000000000

1. 0000000000

2. □□□□□□

3. □□□□□

4.

5.

- [illegible]

□□□□ **4**□□□□□□□□ - □□□□□□

□□□ □□□□□□□□□□ - □□□□□□□□

1.

- □□□□□□□

- Stripe 環境構築“準備”

- 環境構築の準備

2. 環境構築

- 環境構築の準備
- 環境構築の準備 → 環境構築
- 環境構築の準備
- 環境構築

3. 環境構築

- 環境構築 “環境構築” 環境構築
- 環境構築の準備

4. 環境構築

- 環境構築の準備
- 環境構築の準備
- 環境構築 “環境構築” 環境構築
- 環境構築の準備
- 環境構築 “環境構築”
- Stripe 環境構築

5. 環境構築

- 環境構築 “環境構築”
- `payment_reference` 環境構築 Stripe 環境構築 ID

環境構築の準備

環境構築 5環境構築の準備

環境構築の準備 - 環境構築の準備

1. 環境構築

- 環境構築 #INV-2025-001234 環境構築の準備
- 環境構築の準備
- 環境構築

2. 環境構築

- 環境構築 環境構築 → 環境構築
- 環境構築の準備
- 環境構築 (環境構築)

3. □□□□□

4. □□□□□□□□

5. □□□□□□□□

- [illegible]

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

1.

2. □□□□□□□□

3. □□□□□□□□

- 0000 #1234 000000
- 000000
- 00 "00" 0000
- 000000"0000 2025-02-10 - 00 #001"

- 姓名 "张小明"
- 工号 #1234 部门 "研发部"

4. 项目信息

- 项目 #1235 项目名称
 - 项目“智能助手 2025-02-10 - 任务 #001”
- 项目 #1236 项目
 - 项目“智能助手 2025-02-10 - 任务 #001”

5. 个人信息

- 姓名 张三
- 身份证号 123456789012345678
- 手机号 13800138000

6. 财务信息

- 工资 月薪 → 每月工资 3000.00 元
- 报销金额 "1000" 元

备注：以上信息均为系统自动生成，如有错误，请及时联系管理员。

姓名

身份证号

手机号

- 项目 #1235 项目名称
- 项目 #1236 项目
- 项目 #1237 项目
- 项目 #1238 项目

工资

- 工资 月薪 → 每月工资 3000.00 元
- 报销金额 "1000" 元
- 报销金额 "1000" 元
- 报销金额 "1000" 元
- 报销金额 "1000" 元 PDF

📄 目錄

- 關於本書
- 關於本書的POS
- 關於本書的PDF
- 關於 Stripe

📄 目錄

- 關於本書
- 關於本書的PDF
- 關於本書的PDF
- 關於本書的PDF

📄 目錄

📄 目錄

- 關於本書
- 關於本書
- 關於本書

📄 目錄

- 關於本書
- 關於本書
- 關於 PDF
- 關於本書

📄 目錄

- 關於本書
- 關於本書
- 關於本書

📁📁📁📁

📁📁📁📁

- 📁 Mailjet 📁📁📁📁📁📁📁📁
- 📁📁📁 PDF 📁📁📁📁📁
- 📁📁📁📁📁📁 Stripe📁 PayPal📁📁📁📁📁
- 📁📁📁📁📁📁📁

📁📁📁📁

- 📁📁📁📁📁📁📁📁
- 📁 📁📁📁 📁 - 📁📁📁📁📁📁
- 📁 📁📁📁 📁 - 📁📁📁📁
- 📁 📁 📁 📁 📁📁📁📁

📁📁📁

- 📁📁📁📁📁📁📁📁
- 📁📁📁📁📁📁📁📁📁📁📁📁
- 📁📁📁📁📁📁📁

📁❓❓❓📁📁📁📁📁📁

📁📁📁📁📁📁📁📁

📁📁

- 📁📁📁📁📁📁📁📁
- 📁📁📁“📁📁📁📁📁”

📁📁

1. 📁📁📁 → 📁 📁 → 📁📁 📁
2. 📁📁📁📁📁📁📁📁📁
3. 📁📁📁📁📁📁

📋 目錄

- 關於本書
- 如何閱讀本書
- 關於Visa、Mastercard、Amex 等
- 關於本書的更新

📖 第一章 關於本書

📖 1.1

- 本書的用途
- 本書的結構
- 本書的更新

📖 1.2

1. 本書的用途
2. 本書的結構
3. 本書的更新
4. 本書的更新

📖 1.3

- 本書的用途
- 本書的結構
- 本書的更新

📖 第二章 關於本書

📖 2.1

- 本書的用途
- 本書的結構
- 本書的更新

📖 2.2



111

- [illegible]

111

- 000 000000000 "000" 00000000000000
- 000 00 "0000" 00000000000000000000

5/5

- 00000000000000000000/000000000000
- 00000000 ID 00000000

□□□□□

- 0000/0000000000000000
- 00000000 (:) 0000000/000000/0000

--	--	--	--

- `basics_payment` - 基礎的支払
- `payment_system_guide` - 決済 API 利用ガイド
- `payments_transaction` - 決済取引
- `payments_invoices` - 決済請求書
- `integrations_mailjet` - 決済連携 Mailjet
- `csa_activity_log` - 決済履歴
- `csa_rate` - OCS/CGRateS 決済料金

CGRateS 安装指南

本指南将指导您如何安装 CGRateS 并配置 OmniCRM 数据库。

简介

OmniCRM 数据库 (CGRateS) 是一个用于存储和查询呼叫记录数据的数据库。

本指南将指导您如何安装 CGRateS 并配置 OmniCRM 数据库。Playbooks 是用于自动化任务的脚本。

安装

1. 安装 CGRateS 数据库。

2. 安装 OmniCRM 数据库。

3. ID - 安装 ID 数据库。

4. 安装 OmniCRM 数据库。

5. 安装 OmniCRM 数据库。

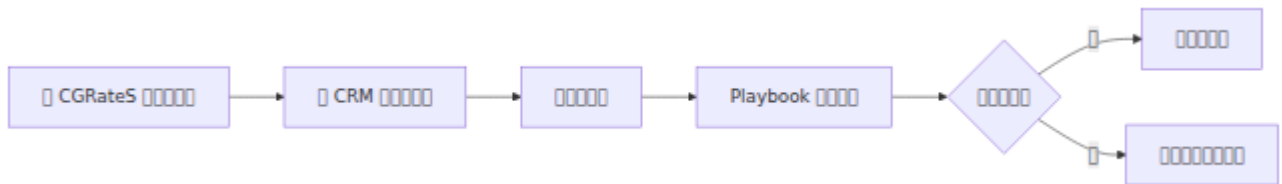
6. 安装 OmniCRM 数据库。

配置

1. 配置 **ExecuteAction** 和 **Playbook** 数据库。

部署

1. 部署 Python 数据库。



□□□□□□□□

□□□□□□□□□□□□

- **CGRateS** □□□ `ActionsId = "Action_50gb-data-pack"`
- **CRM** □□□ `product_slug = "50gb-data-pack"`
- □ **Playbook** □□ `cgr_action_name = "Action_" + product_slug`

□ Playbook □□□□□□ `product_slug` □□□□□□□□□□ `ExecuteAction` □□□□□□□□ `CGRateS` □□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□□□□

1. □□□□□□ - □□□□□□□□
2. □□□□□□ - □□□□□□□□□□□□
3. □□□□□□ - □□ `OCS` □ `CRM` □ `Python` □□

□□□□□□□□□□□□□□

```

import cgrateshttpapi

OCS_Obj = cgrateshttpapi.CGRateS("ocs.example.com", "2080")
tenant = "your_tenant"

# 1. CGRateS
Action_50GB_Data_Pack = {
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_50gb-data-pack",
        "Tenant": tenant,
        "Actions": [
            {
                "Identifier": "*topup",
                "BalanceType": "*data",
                "Units": 50 * 1024 * 1024 * 1024,
                "ExpiryTime": "+720h",
                "Weight": 10
            }
        ]
    }]
}

result = OCS_Obj.SendData(Action_50GB_Data_Pack)
assert result['error'] is None or result['error'] == "EXISTS"

# 2. CRM
# (product_slug = "50gb-data-pack" Action_50gb-data-pack)

```

00000000000000

```

#  Playbook
- name:
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body:
      {
        "method": "APIerSv1.ExecuteAction",
        "params": [{
          "ActionsId": "Action_50gb-data-pack"
        }]
      }
  register: response

#
# response.json.error = "SERVER_ERROR: "
#

```

-
-
-

JSON 数据格式

```
# 用户数据列表
[
  {
    "ID": "Data_5GB_5days_uuid_abc123",
    "data": {
      "usage": 5368709120, # 5GB用量
      "expiry": "2024-12-29",
      "days": 10
    }
  },
  {
    "ID": "Data_10GB_30days_uuid_def456",
    "data": {
      "usage": 10737418240, # 10GB用量
      "expiry": "2025-01-24",
      "days": 10
    }
  }
]
```

CGRateS 数据格式 - 包含用户ID、用量、过期时间、天数

*topup 和 *topup_reset

通过 ID 识别用户

*topup - 充值

*topup 包含以下字段

字段

- 用户ID
- 充值金额
- 充值时间
- 充值来源

□□□

```
# □□□□□
□□□
  ID: "Data_Package__5368709120"
  □: 1073741824 # □□ 1GB
  □□: 2024-12-24□□□ 1 □□

# □□ *topup □□□
□□□
  □□□: "*topup"
  □□□
    ID: "Data_Package__5368709120" # □□ ID - □□□□
    □: 5368709120 # 5GB
    ExpiryTime: "+5d"

# *topup □□□□□
□□□
  ID: "Data_Package__5368709120"
  □: 6442450944 # 6GB□1GB + 5GB □□□
  OriginalValue: 5368709120 # □□□□□□ 5GB
  Value_hr: "6 GB"
  OriginalValue_hr: "5 GB"
  Remaining_hr: "6 GB of 5 GB (1 GB rolled over)"
  □□: 2024-12-29□□□□□ 5 □□
```

□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□
- □□□□□

*topup_reset - □□□□

*topup_reset □□ □□□□□□□□□□□□ ID□

□□□

- □□□□□□ ID □□□□□

- 查詢查詢❓❓查詢
- 查詢查詢查詢
- 查詢查詢查詢查詢查詢

查詢

```
# 查詢查詢
查詢
ID: "Data_Package__5368709120"
: 1073741824 # 1GB
: 2024-12-24查詢 1 查詢

# 查詢 *topup_reset 查詢
查詢
查詢: "*topup_reset"
查詢
ID: "Data_Package__5368709120" # 查詢 ID - 查詢查詢
: 5368709120 # 5GB
ExpiryTime: "+5d"

# *topup_reset 查詢查詢
查詢
ID: "Data_Package__5368709120"
: 5368709120 # 5GB查詢 1GB 查詢查詢
OriginalValue: 5368709120
Value_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
PercentUsed: 0
: 2024-12-29查詢查詢 5 查詢
```

查詢

- 查詢查詢查詢查詢查詢查詢查詢
- 查詢查詢查詢查詢查詢查詢查詢查詢
- 查詢查詢查詢查詢查詢查詢查詢查詢
- 查詢查詢查詢查詢查詢查詢查詢查詢

データ ID の取得方法

データ ID の取得方法について説明します。

データ ID の取得方法

OmniCRM のデータ ID の取得方法は、API を利用して Web UI から取得します。

データ ID の取得

```
{data_id}__{data_name}
```

データ ID の取得

```
# データ100GB
"AU_Data_Domestic__107374182400"
# データ
# - データ"AU_Data_Domestic"のデータ - 100/100GB
# - データ"__"のデータ
# - データ"107374182400"の100GB のデータ
# - UI データ"AU Data Domestic - 100 GB"

# データ3000min
"AU_Voice_Domestic__1800000000000000"
# - データ"AU_Voice_Domestic"のデータ "AU_Voice_Domestic_3000min"
# - データ"1800000000000000"の3000minのデータ
# - UI データ"AU Voice Domestic - 3000 min"

# データ3000msg
"AU_SMS_Domestic__3000"
# - データ"AU_SMS_Domestic"
# - データ"3000"のデータ
# - UI データ"AU SMS Domestic - 3000 msgs"
```

データ ID の取得方法 - データ ID の取得 データ API の取得方法

API の取得方法

OmniCRM API の CGRateS のデータ ID の取得 _hr


```
{
  "BalanceMap": {
    "*data": [
      {
        "ID": "AU_Data_Domestic__107374182400",
        "Value": 53687091200,
        "ExpiryTime": "2025-01-25T23:59:59Z",
        "Weight": 1200,

        // 詳細情報
        "ID_hr": "AU Data Domestic",
        "OriginalValue": 107374182400,
        "OriginalValue_hr": "100 GB",
        "Value_hr": "50 GB",
        "Remaining_hr": "50 GB of 100 GB",
        "PercentUsed": 50,
        "ExpiryTime_hr": "2025年1月25日22時"
      }
    ]
  }
}
```

API 処理

1. ID 処理

```
balance_id = "AU_Data_Domestic__107374182400"
parts = balance_id.split("__")

descriptive_name = parts[0] # "AU_Data_Domestic"
original_size = int(parts[1]) if len(parts) > 1 else None #
107374182400
```

2. 詳細情報取得

```
# 詳細情報取得
id_hr = descriptive_name.replace("_", " ") # "AU Data
Domestic"
```

3. 〇〇〇〇〇〇〇〇〇〇〇〇

```
# 〇〇〇〇〇〇〇〇〇〇〇〇
if balance_type == "*data":
    original_value_hr = convert_bytes_to_gb(original_size) #
    "100 GB"

# 〇〇〇〇〇〇〇〇〇〇〇〇
elif balance_type == "*voice":
    original_value_hr = convert_ns_to_minutes(original_size) #
    "3000 min"

# 〇〇〇〇〇〇〇〇〇〇〇〇
elif balance_type == "*sms":
    original_value_hr = f"{original_size} msgs" # "3000 msgs"
```

4. 〇〇〇〇〇〇〇〇〇

```
if original_size and original_size > 0:
    percent_used = ((original_size - current_value) /
original_size) * 100
```

5. 〇〇〇〇〇〇〇〇〇

```
remaining_hr = f"{current_value_hr} of {original_value_hr}"
# "50 GB of 100 GB"
```

Web UI 〇〇〇

〇〇〇〇〇〇 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇

```
// 電話番号
// ID: "AU_Data_Domestic__107374182400"
// 日: 53687091200

// 残高表示
<BalanceCard>
  <Title>{balance.ID_hr}</Title>          {/* "AU Data Domestic"
*/}
  <Progress value={balance.PercentUsed}> {/* 50% */}
    {balance.Remaining_hr}              {/* "50 GB of 100 GB"
*/}
  </Progress>
  <Expiry>{balance.ExpiryTime_hr}</Expiry> {/* "2025年1月25日22時"
*/}
</BalanceCard>
```

電話番号

1. 電話番号 - 電話番号UI 表示 "50 GB of 100 GB" 表示 "50 GB 残り"
2. 残高表示 - 残高表示UI 表示
3. 残高表示 - 残高 ID 表示
4. 残高表示 - 残高 ***topup** 表示 70 GB 表示 100 GB
 - 残高 ID 表示 "AU_Data_Domestic__107374182400" 表示 100 GB
 - 残高表示 170 GB
 - UI 表示 "170 GB (70 GB rolled over + 100 GB new)"

残高表示 - 残高 **ID**

残高  残高表示UI 表示

$$\# \quad \square - \square\square\square\square + \square\square\square\square\square\square\square\square$$

```
Action_Data_100GB = {
  "Actions": [
    {
      "BalanceId": f"AU_Data_Domestic__{{100 * 1024 * 1024 *
1024}}",
      "Units": 100 * 1024 * 1024 * 1024
    }
  ]
}
```

-

```
Action_Data_100GB = {
  "Actions": [
    {
      "BalanceId": f"AU_Data_Domestic_100GB__{{100 * 1024 *
1024 * 1024}}", # [][]
      "Units": 100 * 1024 * 1024 * 1024
    }
  ]
}
```

- UI

```
Action_Data_100GB = {
    "Actions": [
        {
            "BalanceId": "AU_Data_Domestic", # __size
            "Units": 100 * 1024 * 1024 * 1024
        }
    ]
}
```

$$\square\square\square\square - \square\square\square\square\square$$

□ □

```
# 平衡情報
{
  "BalanceId": "PAYG_Monetary_Balance",
  "BalanceType": "*monetary",
  "Units": 5000 # $50.00
}

# UI 表示情報
# "5000$50.00"
```

1. ID 取得

ID 取得は、UUID を取得する。

コード

```

- name: 充值套餐
  set_fact:
    uuid: "{{ 99999999 | random | to_uuid }}"
    balance_id: "Data_5days__5368709120_{{ uuid[0:8] }}"

- name: 充值 *topup 套餐
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Balance": {
            "ID": "{{ balance_id }}",
            "Value": 5368709120,
            "ExpiryTime": "+5d",
            "Weight": 10
          }
        }]
      }

```

充值套餐 充值 *topup 套餐

```
# 5GB 5GB
1:
ID: "Data_5days__5368709120_a1b2c3d4"
: 5368709120
Value_hr: "5 GB"
OriginalValue_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
: 2024-12-29

2:
ID: "Data_5days__5368709120_e5f6g7h8"
: 5368709120
Value_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
: 2024-12-30

3:
ID: "Data_5days__5368709120_i9j0k1l2"
: 5368709120
Value_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
: 2024-12-31

# 5GB
# UI "Data 5days - 5 GB of 5 GB"
```

2 ID *topup

ID *topup

```

- name: 充值 ID
  set_fact:
    balance_id: "Data_5days__5368709120"

- name: 充值 *topup 充值金额
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Balance": {
            "ID": "Data_5days__5368709120",
            "Value": 5368709120,
            "ExpiryTime": "+5d",
            "Weight": 10
          }
        }]
      }

```

充值 充值金额


```
# 1 5GB 5GB
ID: "Data_5days__5368709120"
: 5368709120
Value_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
PercentUsed: 0
: 2024-12-29
```

```
# 3 1GB 4GB + 5GB
ID: "Data_5days__5368709120"
: 9663676416 # 4GB + 5GB
Value_hr: "9 GB"
OriginalValue_hr: "5 GB"
Remaining_hr: "9 GB (4 GB rolled over + 5 GB new)"
PercentUsed: -80 #
: 2024-12-27 +5
```

3 ID *topup_reset

ID *topup_reset

```

- name: 000000 ID
  set_fact:
    balance_id: "Monthly_Plan__32212254720"

- name: 00 *topup_reset 0000000000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.SetActions",
        "params": [{
          "ActionsId": "Action_Reset_Monthly_Plan",
          "Actions": [{
            "Identifier": "*topup_reset",
            "BalanceType": "*data",
            "Units": 32212254720, # 30GB
            "ExpiryTime": "*monthly",
            "DestinationIds": "*any",
            "BalanceId": "Monthly_Plan__32212254720",
            "Weight": 10
          }]
        }]
      }

```

000 000000000000 30GB0000000000

```
# 1 25 28GB
ID: "Monthly_Plan__32212254720"
: 32212254720
Value_hr: "30 GB"
Remaining_hr: "30 GB of 30 GB"
PercentUsed: 0
: 2024-12-31

# 1 25 28GB
ID: "Monthly_Plan__32212254720"
: 2147483648
Value_hr: "2 GB"
Remaining_hr: "2 GB of 30 GB"
PercentUsed: 93
: 2024-12-31

# 2 1 ActionPlan *topup_reset
ID: "Monthly_Plan__32212254720"
: 32212254720 # 2GB
Value_hr: "30 GB"
Remaining_hr: "30 GB of 30 GB"
PercentUsed: 0
: 2025-01-31
```

--	--	--	--	--

```
CGRateS = CGRateS * 0.0001; % Convert CGRateS to units of 10^-4
```

--	--	--	--	--	--	--	--

⬜⬜⬜⬜❓❓❓ Blocker: true ⬜⬜

1. 00000000 → 0000000000000000
2. 0000000000 → 000000000000000000
3. 0000 → INSUFFICIENT CREDIT BALANCE BLOCKER 0000

5/5

- 0000000000000000 000000000000000000000000
- 0000000000000000000000CGRateS 0000 000000
- 0000000000000000 *voice 0 *data 0 *sms 0 *monetary

--	--	--	--	--	--

1. □□□□

□ □

```

Action_Suspend_Account = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_suspend-account",
        "Overwrite": True,
        "Tenant": tenant,
        "Actions": [
            # 0000000000000000
            {
                "Identifier": "*topup",
                "BalanceId": "Suspension_Blocker",
                "BalanceType": "*monetary",
                "DestinationIDs": "*any",
                "Units": 0, # 00
                "BalanceWeight": 9999, # 00000 - 00000
                "Blocker": True, # 000000
                "Weight": 10
            }
        ]
    }]
}

result = OCS_Obj.SendData(Action_Suspend_Account)

```

0000 0000/00/SMS 0000000000000000

2. □□□□

Configuration

```
Action_Monthly_Plan_With_Cap = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_monthly-with-cap",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # 10GB plan
      {
        "Identifier": "*topup_reset",
        "BalanceId": f"Included_Data__{10 * 1024 * 1024 *
1024}",
        "BalanceType": "*data",
        "DestinationIDs": "Dest_PLMN_OnNet",
        "Units": 10 * 1024 * 1024 * 1024,
        "ExpiryTime": "*month",
        "BalanceWeight": 1200, # 
        "Weight": 95
      },
      # $50 
      {
        "Identifier": "*topup_reset",
        "BalanceId": "Overage_Cap",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any",
        "Units": 5000, # $50.00 
        "ExpiryTime": "*month",
        "BalanceWeight": 1000, # 
        "Blocker": True, # $50 
        "Weight": 90
      }
    ]
  }
}
```

Notes

1. 10GB plan → Included_Data_10GB
2. 5GB → Overage_Cap PAYG

3. Overage_Cap 0 \$0 →

3.

```
Action_Trial_Account = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_trial-100-minutes",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # 100  - 
      {
        "Identifier": "*topup",
        "BalanceId": f"Trial_Voice__{100 * 60 *
10000000000}",
        "BalanceType": "*voice",
        "DestinationIDs": "Dest_Domestic_All",
        "Units": 100 * 60 * 10000000000, # 100
        "ExpiryTime": "+720h", # 30
        "BalanceWeight": 1200,
        "Blocker": True, # 100 
        "Weight": 10
      }
    ]
  }]
}
```

100

4.

```

Action_Block_Premium_Numbers = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_block-premium",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # 普通充值
      {
        "Identifier": "*topup_reset",
        "BalanceId": "Regular_Usage",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any",
        "Units": 10000, # $100
        "ExpiryTime": "*month",
        "BalanceWeight": 1000,
        "Weight": 20
      },
      # 充值0900 普通
      {
        "Identifier": "*topup",
        "BalanceId": "Premium_Blocker",
        "BalanceType": "*monetary",
        "DestinationIDs": "Dest_Domestic_Premium",
        "Units": 0, # 0
        "BalanceWeight": 2000, # 2000 - 10000
        "Blocker": True,
        "Weight": 10
      }
    ]
  }]
}

```

普通

- 普通 → 普通 Regular_Usage 普通
- 充值0900 → 普通 Premium_Blocker 2000 > 1000 → 普通

○○○○○○○○○○○○○○

○○○ Blocker ○ Disabled ○○○

○○	○○○	○○
○○	○○○○○○○○○○○○	○○○○○○○
○○ > 0	○○○○○○○	○○○○○○/○○
○○ = 0	○○○○○○○○○○	○○○○○○○○○○○○○○○○
○○	○○○○○○○○○○○○○○	○○○○○○○○○

```
# ○○○○○○ 10GB○○○○○○○○○○
{
  "BalanceId": f"Data_Cap_{10 * 1024 * 1024 * 1024}",
  "Units": 10 * 1024 * 1024 * 1024,
  "Blocker": True # ○ 10GB ○○○○
}

# ○○○○○○○○○○○○○○○○○○○
{
  "BalanceId": f"Bonus_Data_{5 * 1024 * 1024 * 1024}",
  "Units": 5 * 1024 * 1024 * 1024,
  "Disabled": True # ○○○○○○○○○
}
```

○○○○○○○○○○○○○○○○

○○○○○○○○○○○○○○○○○○


```

Action_Safe_Hybrid_Plan = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_safe-hybrid-plan",
        "Overwrite": True,
        "Tenant": tenant,
        "Actions": [
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },
            # 500 语音
            {
                "Identifier": "*topup_reset",
                "BalanceId": f"Domestic_Voice__{500 * 60 *
10000000000}",
                "BalanceType": "*voice",
                "DestinationIDs": "Dest_Domestic_All",
                "Units": 500 * 60 * 10000000000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 95
            },
            # $20 话费
            {
                "Identifier": "*topup_reset",
                "BalanceId": "Overage_Allowance",
                "BalanceType": "*monetary",
                "DestinationIDs": "*any",
                "Units": 2000, # $20.00
                "ExpiryTime": "*month",
                "BalanceWeight": 1000,
                "Weight": 90
            },
            # $50 话费
            {
                "Identifier": "*topup_reset",
                "BalanceId": "Hard_Spending_Cap",
                "BalanceType": "*monetary",
                "DestinationIDs": "*any",
                "Units": 5000, # $50.00
                "ExpiryTime": "*month",

```

```

        "BalanceWeight": 500, # 0000 - 0000
        "Blocker": True, # 00000000
        "Weight": 85
    }
}
}

```

00000

1. **0-500**000 00 Domestic_Voice_500min0000
2. **500-700**000 00 Overage_Allowance0000 \$0.10 = \$200200000
3. **700-1200**000 00 Hard_Spending_Cap0000 \$0.10 = \$500500000
4. 0**1200**0000 Hard_Spending_Cap 00 → 00000000

000000 1200 00000000 \$500

0000000000

1. 0000000000

```

"BalanceWeight": 9999 # 0000000000

```

2. 000000000000

```

"Units": 0, # 0000
"Blocker": True

```

3. 000000000000

- 00 ActionTriggers 0 80%090%0100% 0000000000
- 000000000000000000

4. 000000000000

```
# 调用 *remove_balance 接口
{
  "Identifier": "*remove_balance",
  "BalanceId": "Suspension_Blocker"
}
```

5. 配置项

- 配置项 `INSUFFICIENT_CREDIT_BALANCE_BLOCKER` 为
- 调用 CDR 接口 = -1.0 配置
- 配置项

配置项

配置项

配置项

- 配置项
- DestinationIDs 配置项
- 配置项 `True`

配置项

```
# 配置项
OCS_Obj.SendData({
  'method': 'ApierV2.GetAccount',
  'params': [{"Tenant": tenant, "Account": "service_uuid"}]
})

# 配置
# - Blocker: true
# - BalanceWeight 配置项9999
# - DestinationIDs 配置项
# - = 0
```

配置项

配置项 配置项

Item 1:
ID: "Premium_Data"
Size: 5GB
Count: 20

Item 2:
ID: "Standard_Data"
Size: 10GB
Count: 10

Item 1 count (20) > Item 2 count (10)
Item 2 size (10GB) > Item 1 size (5GB)

Item 1 is larger than Item 2

Item 3

Item 3 is the largest

Item 1:
ID: "Data_Package_A"
Size: 5GB
Valid Until: 2024-12-25
Count: 10

Item 2:
ID: "Data_Package_B"
Size: 10GB
Valid Until: 2025-01-15
Count: 10

Item 2 is larger and valid longer

```
# 数据包的元数据
包 1:
  ID: "Data_Package_A"
  大小: 5GB
  创建时间: 2024-12-25
  索引: 11 # 索引值 = 偏移量

包 2:
  ID: "Data_Package_B"
  大小: 10GB
  创建时间: 2025-01-15
  索引: 10 # 索引值 = 偏移量

# 包 1 的索引值 11 > 包 2 的索引值 10
```

数据包元数据

包 1

包 1 的元数据

包 1 的元数据“5GB 5”

包 1

```

- name: 新增 UUID 数据表
  set_fact:
    uuid: "{{ 99999999 | random | to_uuid }}"

- name: 新增 5GB 数据
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Categories": "*any",
          "Balance": {
            "ID": "Data_5GB_{{ uuid[0:8] }}",
            "Value": 5368709120,
            "ExpiryTime": "+120h", # 5天
            "Weight": 10
          }
        }]
      }

```

数据表

- 12月24日数据表 → 新增 5GB 2024 年 12 月 29 日数据
- 12 月 25 日数据表 → 新增 5GB 2024 年 12 月 30 日数据
- 数据表数据量限制为 10 万条数据

新增 2 数据表

新增 “新增 5GB 数据”数据表数据表数据表

数据

```
- name: 充值接口
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.AddBalance",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "BalanceType": "*data",
        "Balance": {
          "ID": "Rollover_Monthly_50GB",
          "Value": 53687091200, # 50GB
          "ExpiryTime": "+720h", # 30天
          "Weight": 10
        }
      }]
    }
}
```

充值接口 *topup 充值接口

充值接口

- 1 充值 50GB 30 天
- 20 充值 30GB 20GB
- 20 充值 → 70GB 20GB + 50GB 20 天 + 30 天

3 充值接口

“100GB 天” 100GB 天

天


```
- name: 设置重置动作
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.SetActions",
      "params": [{
        "ActionsId": "Action_Monthly_100GB_Reset",
        "Overwrite": true,
        "Actions": [{
          "Identifier": "*topup_reset",
          "BalanceType": "*data",
          "Units": 107374182400, # 100GB
          "ExpiryTime": "*monthly",
          "BalanceId": "Monthly_Plan__107374182400",
          "Weight": 10
        }]
      }]
    }
```

```
- name: 设置动作计划
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.SetActionPlan",
      "params": [{
        "Id": "ActionPlan_Monthly_100GB",
        "ActionPlan": [{
          "ActionsId": "Action_Monthly_100GB_Reset",
          "Time": "*monthly",
          "Weight": 10
        }],
        "Overwrite": true,
        "ReloadScheduler": true
      }]
    }
```

```
- name: 设置动作计划 权重
```



```
# 000000000000
- name: 0000000000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Balance": {
            "ID": "Bonus_Data",
            "Value": 5368709120, # 5GB
            "ExpiryTime": "+240h", # 10
            "Weight": 20
          }
        }]
      }
}
```

```
# 000000000000
- name: 0000000000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Balance": {
            "ID": "Regular_Data",
            "Value": 53687091200, # 50GB
            "ExpiryTime": "+720h", # 30
            "Weight": 10 # 0000
          }
        }]
      }
}
```

套餐

- 5GB 套餐 20+ 50GB 套餐 10
- 套餐 5GB 套餐
- 套餐 50GB 套餐
- 套餐

套餐

CGRateS 套餐

套餐

- *topup** - 套餐
- *topup_reset** - 套餐
- *debit** - 套餐
- *debit_reset** - 套餐
- *reset_account** - 套餐

套餐

OmniCRM 套餐

套餐 1

- 套餐 → 套餐 ID 套餐 UUID
- 套餐 → 套餐 ID 套餐 *topup_reset
- 套餐 → 套餐 ID 套餐 *topup

套餐 2

- 套餐 → 套餐 *topup 套餐
- 套餐 → 套餐 *topup_reset 套餐
- 套餐 → 套餐 ID

例 3 課金プランの定義

課金プラン → 課金プランID/課金プラン名

課金プラン → 課金プランID = 課金プラン名

課金プラン → 課金プランID30課金プラン20課金プラン10課金プラン

課金プラン/課金プラン → 課金プランID/課金プラン名

例 4 課金プランの定義

課金プラン → 課金プランID+720h 30 課金プラン

課金プラン → 課金プランID ActionPlan 課金プラン *monthly

課金プラン → 課金プランID *unlimited 課金プランID

CGRateS 課金プラン Playbooks

課金プランID

```

- name: CGRateS
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.SetActions",
        "params": [{
          "ActionsId": "Action_Name_Here",
          "Overwrite": true, #
          "Actions": [
            {
              "Identifier": "*topup", # *topup_reset
              "BalanceType": "*data", # *data[*voice[*sms[*monetary
              "Units": 5368709120, #
              "ExpiryTime": "+120h", # +Xh[*unlimited[*monthly
              "DestinationIds": "*any", # *any
              "BalanceId": "Balance_Name", #
              "Weight": 10, # =
              "Blocker": false, # true
              "Disabled": false, # true
              "SharedGroups": "" #
            },
            {
              "Identifier": "*cdrlog", # CDR
              "BalanceType": "*generic",
              "ExtraParameters": "{\\"Category\\":\\"^activation\\",\\"Destination\\":\\"Addon Name\\"}"
            }
          ]
        }]
      }

```

CGRateS

ActionsId -

Identifier - *topup *topup_reset *cdrlog

BalanceType - 平衡種類

- `*data` - データ
- `*voice` - 音声
- `*sms` - SMS
- `*monetary` - 金銭
- `*generic` - 汎用

Units - 単位/単位数

ExpiryTime - 有効期限

- `+Xh` - 時間 `+720h` = 30 時間
- `*unlimited` - 無制限
- `*monthly` - 毎月
- `2024-12-31T23:59:59Z` - 日付

BalanceId - 残高 ID = 残高 ID = 残高

Weight - 重み = 重み

Blocker - ブロック true = ブロック

Disabled - 無効 true = 無効

Python 接続

Python 接続 `cgrateshttpapi` 接続
`OCS_Obj.SendData()` 接続

準備

```
import cgrateshttpapi
import time

OCS_obj = cgrateshttpapi.CGRateS("ocs.example.com", "2080")
tenant = "your_tenant_name"
tpid = str(tenant) + "_" + str(int(time.time()))
```

送信準備

送信準備の準備

送信準備の準備

- 送信準備 - 送信/SMS 送信準備の準備 `Dest_International_UK`
- **PLMN** 送信 - 送信準備の準備 `Dest_PLMN_OnNet` `Dest_PLMN_US_Verizon`

送信準備

- 送信/SMS 送信 → 送信準備の準備 TO
- 送信 → 送信 PLMN 送信準備の準備 FROM

送信準備の準備

- 送信準備の準備の準備
- PLMN 送信準備の準備
- PLMN 送信準備の準備
- 送信準備の準備

送信準備 **CGRateS** 送信準備

送信

送信準備の準備


```
# 1GB의 크기 (1024 * 1024 * 1024 바이트)
1_GB = 1 * 1024 * 1024 * 1024 # 1073741824 바이트
100_GB = 100 * 1024 * 1024 * 1024

# 1분, 3000분 (30분)의 크기
1_minute = 60 * 1000000000 # 60 초
3000_minutes = 3000 * 60 * 1000000000

# 3000개의 SMS
3000_sms = 3000
```

1 Python

```

Action_AU_Premium_Plan_1 = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_au-premium-plan-1",
        "Overwrite": True,
        "Tenant": str(tenant),
        "Actions": [
            # 重置账户
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },
            # 100GB 国内流量
            # 国内流量 PLMN 重置
            # 国内流量 PLMN
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Data_Domestic__" + str(100 * 1024
* 1024 * 1024),
                "BalanceType": "*data",
                "DestinationIDs": "Dest_PLMN_OnNet", # 国内 PLMN
                #mcc505.mnc057
                "Units": 100 * 1024 * 1024 * 1024,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 90
            },
            # 3000 国内语音
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Voice_Domestic__" + str(3000 * 60
* 1000000000),
                "BalanceType": "*voice",
                "DestinationIDs":
                "Dest_AU_Mobile;Dest_AU_Fixed;Dest_AU_TollFree;",
                "Units": 3000 * 60 * 1000000000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 89
            },
            # 3000 国内
            {

```

```

        "Identifier": "*topup_reset",
        "BalanceId": "AU_SMS_Domestic__" + str(3000),
        "BalanceType": "*sms",
        "DestinationIDs": "Dest_AU_Mobile;",
        "Units": 3000,
        "ExpiryTime": "*month",
        "BalanceWeight": 1200,
        "Weight": 88
    },
    # 6GB 流量包
    {
        "Identifier": "*topup_reset",
        "BalanceId": "AU_Roaming_Data__" + str(6 * 1024 *
1024 * 1024),
        "BalanceType": "*data",
        "DestinationIDs": "Dest_Roaming_All",
        "Units": 6 * 1024 * 1024 * 1024,
        "ExpiryTime": "*month",
        "BalanceWeight": 1100,
        "Weight": 87
    },
    # 流量 CDR
    {
        "Identifier": "*cdrlog",
        "BalanceType": "*generic",
        "ExtraParameters": "
{"Category\":"^activation\","Destination\":"AU Premium Plan
1\}"
    }
]
}]
}

# 流量 CGRateS
result = OCS_Obj.SendData(Action_AU_Premium_Plan_1)
assert result['error'] is None or result['error'] == "EXISTS"
print("流量Action_au-premium-plan-1")

```

流量

- 流量 *reset_account 流量
- 流量 *topup_reset 流量

- BalanceWeight 1200 > 1100
- Weight
- `*cdrlog`

2Python

20GB

```

Action_AU_Data_Addon_20GB = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_au-data-addon-20gb",
        "Overwrite": True,
        "Tenant": str(tenant),
        "Actions": [
            # 重置账户
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },
            # 20GB 国内流量
            # 重置 20GB 国内流量 PLMN 重置国内流量
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Data_Domestic__" + str(20 * 1024
* 1024 * 1024),
                "BalanceType": "*data",
                "DestinationIDs": "Dest_PLMN_OnNet", # 国内 PLMN
                #mcc505.mnc057
                "Units": 20 * 1024 * 1024 * 1024,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 90
            }
        ]
    }]
}

result = OCS_Obj.SendData(Action_AU_Data_Addon_20GB)
assert result['error'] is None or result['error'] == "EXISTS"
print("重置Action_au-data-addon-20gb")

```

3 Python

```

Action_AU_International_Voice_100min = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_au-international-voice-100min",
        "Overwrite": True,
        "Tenant": str(tenant),
        "Actions": [
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },
            # 100 1000000000
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Voice_International__" + str(100
* 60 * 10000000000),
                "BalanceType": "*voice",
                "DestinationIDs": "Dest_International_All",
                "Units": 100 * 60 * 10000000000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1000,
                "Weight": 90
            }
        ]
    }]
}

result = OCS_Obj.SendData(Action_AU_International_Voice_100min)
assert result['error'] is None or result['error'] == "EXISTS"
print("10000000000 Action au-international-voice-100min")

```

/SMS PLMN

Python

--	--	--	--	--	--	--

- **ActionsId** - CRM product_slug
- **Overwrite** - True ID
- **Tenant** - CGRateS

- **Actions** - 設定可能なアクション

設定可能なアクション

- **Identifier** - 識別子 *topup *topup_reset *cdrlog
- **BalanceId** - 残高ID
- **BalanceType** - 残高タイプ *data *voice *sms *monetary
- **DestinationIDs** - 宛先ID
 - 宛先 /SMS 宛先
 - 宛先 "Dest_AU_Mobile" "Dest_International_UK"
 - 宛先 PLMN 宛先
 - 宛先 "Dest_PLMN_OnNet" "Dest_PLMN_US_Verizon"
- **Units** - 単位
- **ExpiryTime** - 有効期限 *month +720h 2024-12-31
- **BalanceWeight** - 残高重み = 10000
- **Weight** - 重み = 10000
- **Blocker** - ブロッカー True
- **Disabled** - 無効 True

設定可能なアクション

設定可能なアクション

```
- name: 00000000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "APIerSv1.ExecuteAction",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "ActionsId": "Action_Name_Here"
        }]
      }
}
```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

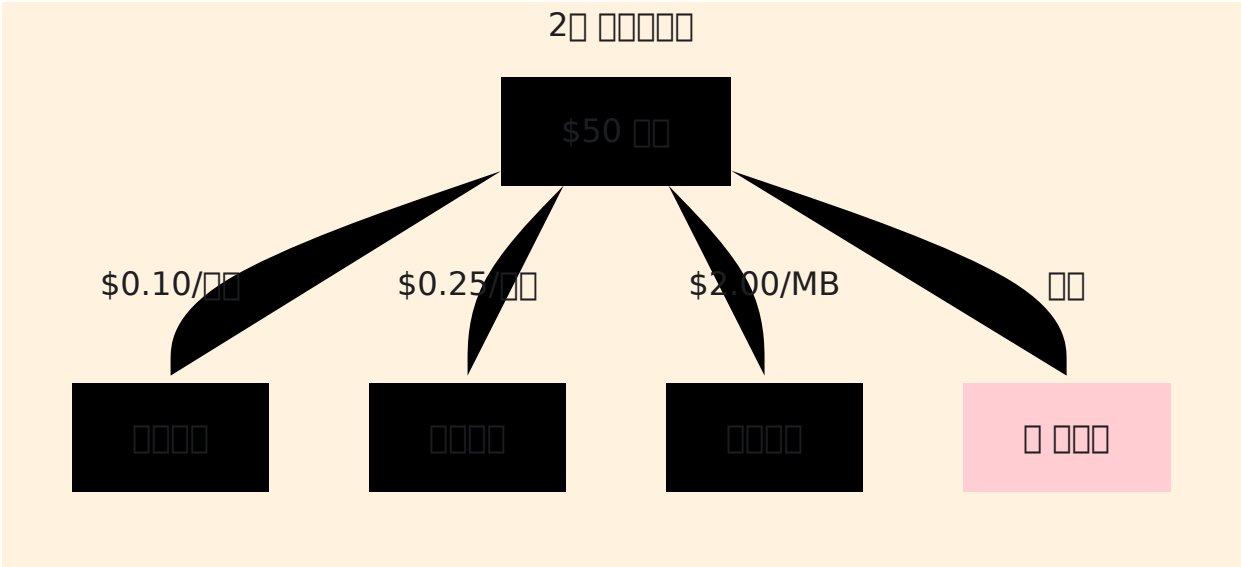
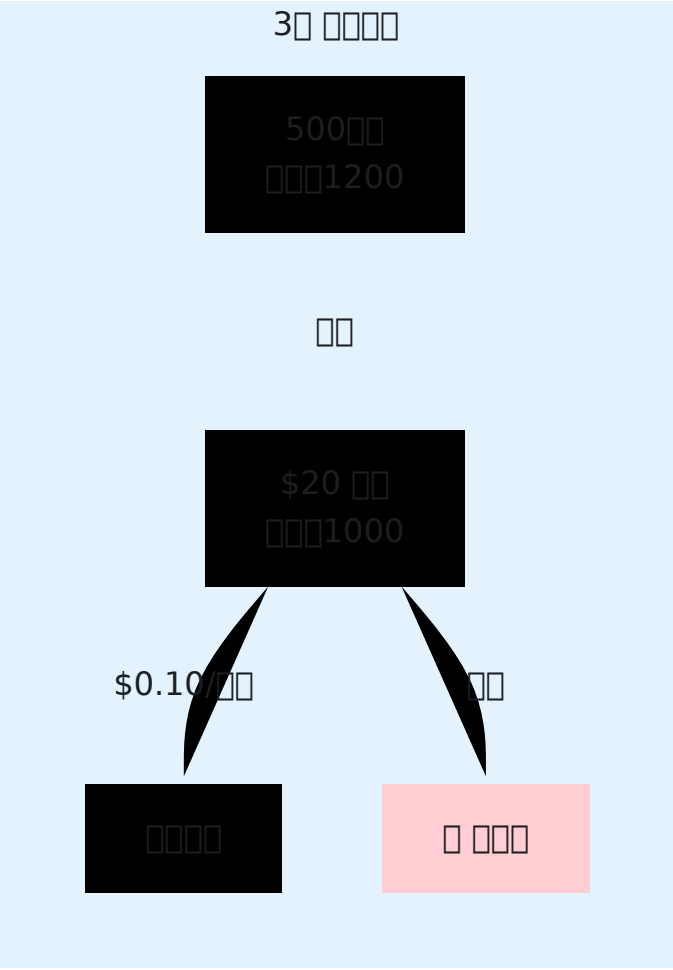
--	--	--	--	--	--

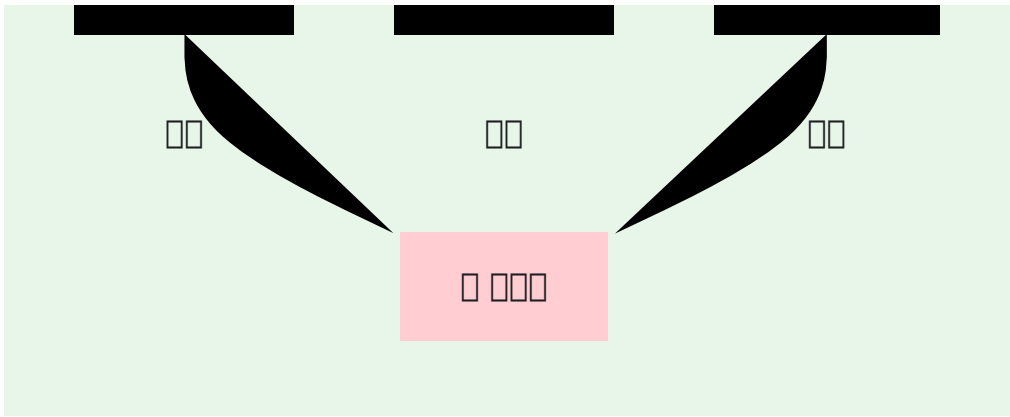
[illegible]

□□□□□ vs □□□ vs □□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

項目	単位	単位 PAYG	単位
総売上	円 総売上	円 総売上	△ 総売上
総売上	円 総売上	円 総売上/円	円 円 + 総売上
総売上	円 総売上	円 総売上	円 総売上
総売上	円 総売上	円 総売上	△ 総売上
総売上	△ 総売上	円 総売上	円 総売上
総売上	△ 総売上 ARPU	円 円 ARPU 総売上	円 総売上 ARPU
総売上	円 総売上	円 総売上	円 総売上
総売上	総売上	総売上	総売上





Server 1 100GB

Server 2 100GB Server 3 100GB RAID 10 200GB

RAID 10 200GB

```

Action_Domestic_Plan = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_domestic-plan",
        "Overwrite": True,
        "Tenant": tenant,
        "Actions": [
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },
            # 500MB
            {
                "Identifier": "*topup_reset",
                "BalanceId": f"Domestic_Voice__{500 * 60 *
10000000000}",
                "BalanceType": "*voice",
                "DestinationIDs": "Dest_Domestic_All", # 
                "Units": 500 * 60 * 10000000000, # 500MB
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 90
            },
            # 1000MB
            {
                "Identifier": "*topup_reset",
                "BalanceId": "Domestic_SMS__1000",
                "BalanceType": "*sms",
                "DestinationIDs": "Dest_Domestic_All",
                "Units": 1000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 89
            },
            # 10GB 
            {
                "Identifier": "*topup_reset",
                "BalanceId": f"Domestic_Data__{10 * 1024 * 1024 *
1024}",
                "BalanceType": "*data",
                "DestinationIDs": "Dest_PLMN_OnNet", # PLMN
                "Units": 10 * 1024 * 1024 * 1024,

```

```

        "ExpiryTime": "*month",
        "BalanceWeight": 1200,
        "Weight": 88
    },
    {
        "Identifier": "*cdrlog",
        "BalanceType": "*generic",
        "ExtraParameters": "
{\\\"Category\\\":\\\"^activation\\\",\\\"Destination\\\":\\\"Domestic Plan\\\"}\",
        "Weight": 80
    }
]
}]
}

result = OCS_Obj.SendData(Action_Domestic_Plan)
assert result['error'] is None or result['error'] == "EXISTS"

```

□□□□□

- □□□□□1-555-1234□ → □□ 500 □□□□
- □□□□□44-20-xxx□ → □□□□□□□□□□□□□□□□□□□□
- □□□□ → □□ 1000 □□□□
- □□□□□□ → □□ 10GB □□
- □□□□ → □□□□□□□□□□□□□□□□□□□□

□□□

- □□□□□□□□□□
- □□□□□□□
- □□□□□□

□□□

- □□□□□□ - □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□

2 PAYG

CGRateS

PAYG/

\$50 PAYG

```
Action_PAYG_Credit = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_payg-50-credit",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # $50
      {
        "Identifier": "*topup",
        "BalanceId": "PAYG_Monetary_Balance",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any", #
        "Units": 5000, # $50.00
        "ExpiryTime": "+2160h", # 90
        "BalanceWeight": 1000, #
        "Weight": 90
      },
      {
        "Identifier": "*cdrlog",
        "BalanceType": "*generic",
        "ExtraParameters": "
{\\\"Category\\\":\\\"^activation\\\",\\\"Destination\\\":\\\"$50 PAYG
Credit\\\"}\",
        "Weight": 80
      }
    ]
  }]
}

result = OCS_Obj.SendData(Action_PAYG_Credit)
```

PAYG

☐☐ **1**☐☐☐☐☐☐**10**☐☐

- ☐☐☐\$0.10/☐☐
- ☐☐☐ $10 \times \$0.10 = \1.00
- ☐☐☐\$49.00

☐☐ **2**☐☐☐☐☐☐☐☐**5**☐☐

- ☐☐☐\$0.25/☐☐ + \$0.05 ☐☐☐
- ☐☐☐☐ $5 \times \$0.25☐ + \$0.05 = \$1.30$
- ☐☐☐\$47.70

☐☐ **3**☐☐ **Verizon** ☐☐☐☐☐ **100MB** ☐☐

- ☐☐

CGRateS

CGRateS is a free CRM tool for mobile operators - it helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data. It is a free tool for mobile operators to manage their customers and their data.



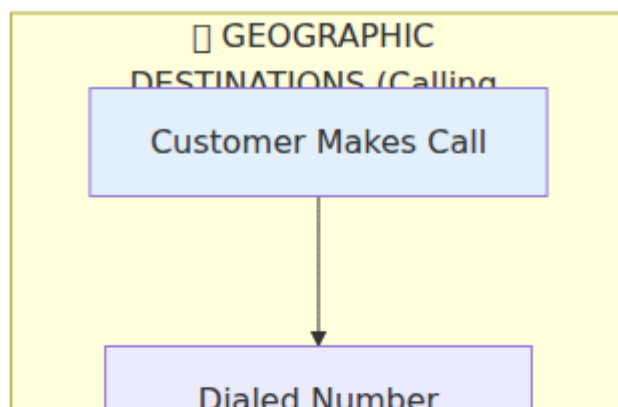
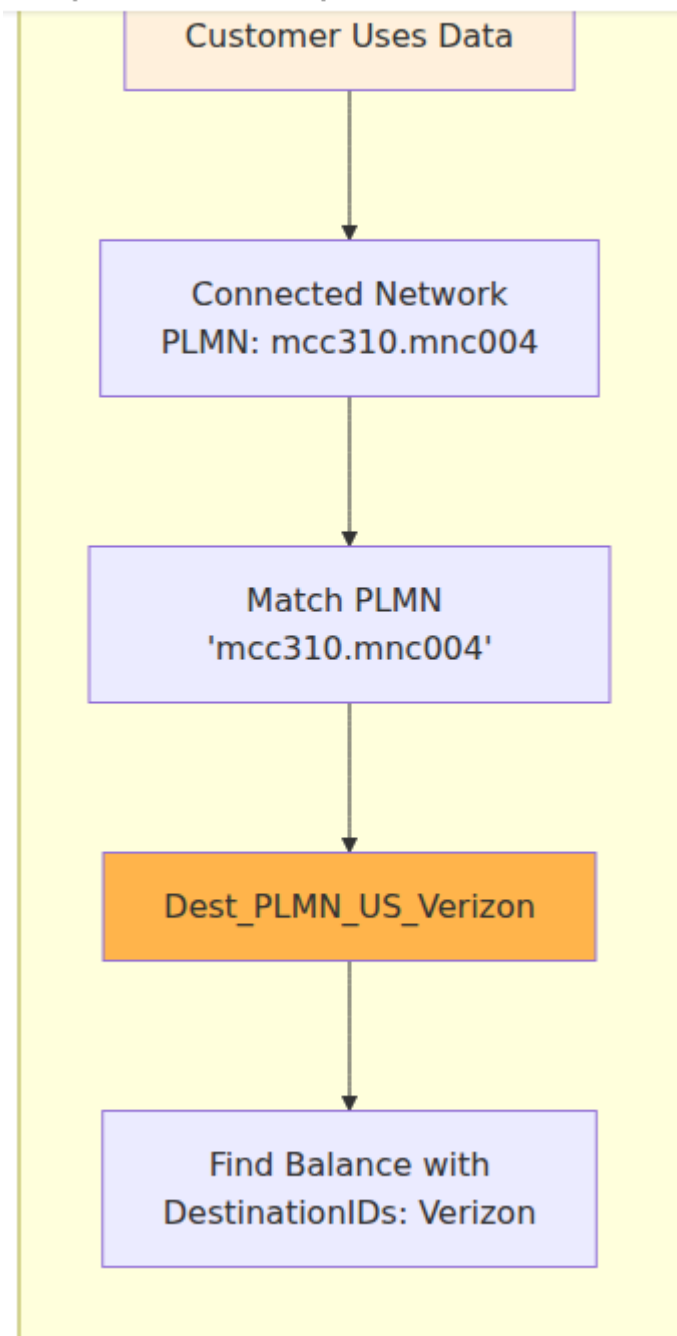
CGRateS is a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.

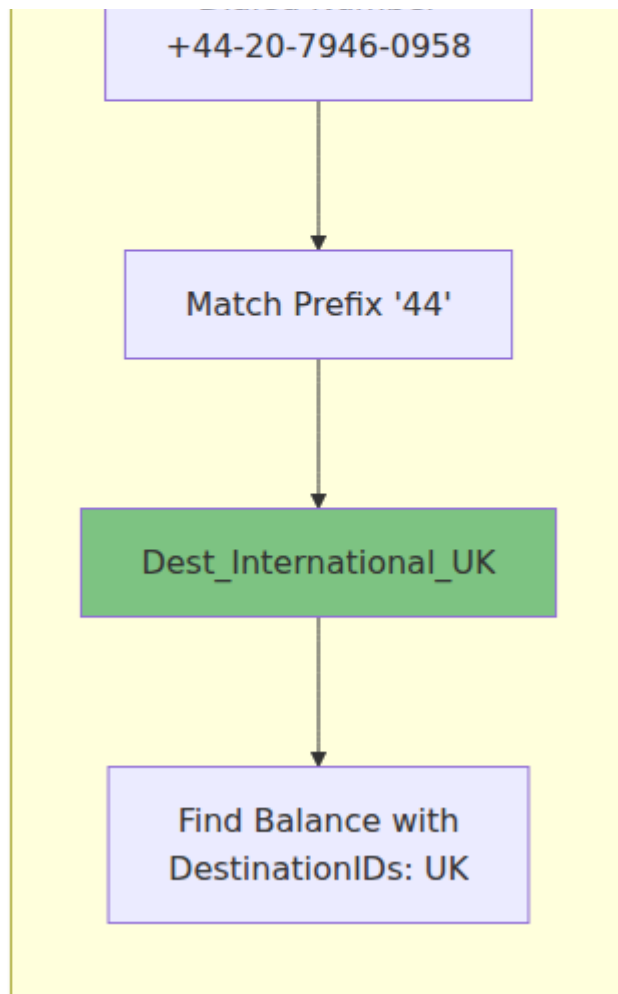
- **CGRateS** - a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.
- **PLMN** - a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.

CGRateS is a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.

- **CGRateS** - a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.
- **PLMN** - a free CRM tool for mobile operators. It helps you to manage your customers and their data. It is a free tool for mobile operators to manage their customers and their data.

OmniCharge OmniRAN Downloads





□□□□

- □□/SMS □□ → □□□□□□□□□□□□
- □□□□ → □□ PLMN □□□□□□□□□□□□

□□□□

```
import cgrateshttpapi
import time

OCS_obj = cgrateshttpapi.CGRateS("ocs.example.com", "2080")
tenant = "your_tenant_name"
tpid = str(tenant) + "_" + str(int(time.time()))
```

1

SMS

□□□□□

```
#
=====
# DOMESTIC DESTINATIONS (US/Canada - NANP)
#
=====

# US/Canada Mobile & Fixed (share area codes)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Domestic_All",
        "Prefixes": ["1"] # NANP (US/Canada)
    }]
})

# US Toll-Free
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Domestic_TollFree",
        "Prefixes": ["1800", "1888", "1877", "1866", "1855",
"1844", "1833"]
    }]
})

# US Premium Rate
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Domestic_Premium",
        "Prefixes": ["1900"]
    }]
})
```

□□□□□

```
#
=====
# INTERNATIONAL DESTINATIONS (By Country)
#
=====

# United Kingdom
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_International_UK",
        "Prefixes": ["44"]
    }]
})

# China
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_International_China",
        "Prefixes": ["86"]
    }]
})

# Australia - Mobile
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_AU_Mobile",
        "Prefixes": ["614"] # Australian mobiles (04xx dialed as
614xx)
    }]
})

# Australia - Fixed Line
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
```

```

        "TPid": tpid,
        "ID": "Dest_AU_Fixed",
        "Prefixes": [
            "612", # NSW (Sydney)
            "613", # VIC (Melbourne)
            "617", # TAS (Hobart)
            "618"  # SA, WA, NT (Adelaide, Perth, Darwin)
        ]
    }]
})

# Australia - Toll-Free
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_AU_TollFree",
        "Prefixes": [
            "611800", # Toll-free
            "611300"  # Local rate
        ]
    }]
})

# Australia - Premium Rate
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_AU_Premium",
        "Prefixes": ["6119"] # Premium rate services
    }]
})

# Europe (grouped)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_International_Europe",
        "Prefixes": [
            "33", # France
            "49", # Germany
            "39", # Italy

```

```

        "34", # Spain
        "31", # Netherlands
        "32", # Belgium
        "41"  # Switzerland
    ]
}

})

# International - All (catch-all, excludes domestic)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_International_All",
        "Prefixes": ["2", "3", "4", "5", "6", "7", "8", "9"] #
    }
    All except "1" (NANP)
    }]
})

```

111

- 国内“国内/国内” → Dest_Domestic_All
- 国际“100国际” → Dest_International_UK
- 澳洲“3000国际澳大利亚” → Dest_AU_Mobile, Dest_AU_Fixed, Dest_AU_TollFree
- 国际“50国际” → Dest International All

2 PLMN - /

PLMN mncXXX.mncYYY

- **MCC** = 0000003 0000
- **MNC** = 00000002-3 0000

PLMN

- `"mcc310.mnc004"` → Verizon 004
- `"mcc310"` →

- `mcc310.mnc004` `mcc234.mnc015` `mcc505.mnc057`

□□□□□□□□□□

□□□□□□□□□□ SIM □□□□□□□□□□

```
# On-Net (Your Home Network)
# This is YOUR operator's PLMN - the network where YOUR customers'
# SIMs are active
# All domestic on-net data balances should use this destination

OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_OnNet",
        "Prefixes": ["mcc505.mnc057"] # Example: MCC 505, MNC 57
        if this is your home network
    }]
})
# Replace mcc505.mnc057 with YOUR actual operator's PLMN code
# All domestic on-net data balances should use the prefixes set
# out here
# Otherwise they're considered roaming
```


PLMN

```
#
=====
# US ROAMING PLMN DESTINATIONS
#
=====

# Verizon Wireless
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_US_Verizon",
        "Prefixes": [
            "mcc310.mnc004", "mcc310.mnc010", "mcc310.mnc012",
            "mcc310.mnc013",
            "mcc311.mnc480", "mcc311.mnc481", "mcc311.mnc482",
            "mcc311.mnc483"
        ]
    }]
})

# All US PLMNs (catch-all)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_US_All",
        "Prefixes": ["mcc310", "mcc311", "mcc312", "mcc313",
            "mcc316"]
    }]
})

# Note: This matches ANY network with these MCCs (310, 311, etc.)
# Customer on mcc310.mnc004 (Verizon), mcc310.mnc410 (AT&T),
# mcc311.mnc580 (US Cellular)
# would ALL match this destination
```

PLMN

```
#
=====
# UK ROAMING PLMN DESTINATIONS
#
=====

# Vodafone UK
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_UK_Vodafone",
        "Prefixes": ["mcc234.mnc015"]
    }]
})

# EE (Everything Everywhere)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_UK_EE",
        "Prefixes": ["mcc234.mnc030", "mcc234.mnc033"]
    }]
})

# All UK PLMNs
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_UK_All",
        "Prefixes": ["mcc234"]
    }]
})
```

“”“”

```

#
=====
# ROAMING ZONES (Multi-Country Groups)
#
=====

# Zone 1: North America
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_NorthAmerica",
        "Prefixes": [
            "mcc310", "mcc311", "mcc312", "mcc313", "mcc316", #
USA
            "mcc302", "mcc334" # Canada
        ]
    }]
})
# Using MCC-only prefixes provides coverage on ANY network in these
countries
# Great for "North America roaming" packages that work on all
carriers

# Zone 2: Europe
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_Europe",
        "Prefixes": [
            "mcc234", # United Kingdom
            "mcc208", # France
            "mcc262", # Germany
            "mcc222", # Italy
            "mcc214", # Spain
            "mcc228" # Switzerland
        ]
    }]
})

# Zone 3: Asia Pacific
OCS_Obj.SendData({

```

```

        'method': 'ApierV2.SetTPDestination',
        'params': [{
            "TPid": tpid,
            "ID": "Dest_PLMN_Zone_AsiaPacific",
            "Prefixes": [
                "mcc505", # Australia
                "mcc460", # China
                "mcc454", # Hong Kong
                "mcc440", # Japan
                "mcc520", # Thailand
                "mcc525", # Singapore
                "mcc530" # New Zealand
            ]
        }]
    })

# Catch-all for any roaming (use with caution)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Roaming_All",
        "Prefixes": ["mcc"] # Matches ANY PLMN (very broad)
    }]
})

```

□□□

- □□□□□□□□“□□ Verizon □□□ 5GB” → Dest_PLMN_US_Verizon
- □□□□□□“□□□□□□□□□□□□ 10GB” → Dest_PLMN_US_All
- □□□□□□“□□□□□ 15GB” → Dest_PLMN_Zone_Europe
- □□□□□□“□□□□□□□□□ 1GB” → Dest_Roaming_All

--	--	--	--	--	--	--	--

項目	項目	項目	項目
項目	項目/項目項目項目	“100項目項目項目項目”	項目SMS
項目 PLMN	項目項目項目項目項目項目	“100GB 項目項目”	項目
項目 PLMN	項目項目項目項目項目項目	“項目 Verizon 項目 5GB”	項目項目項目項目/SMS項目
項目 PLMN	項目項目項目項目項目項目	“項目項目項目項目項目項目 5GB”	項目項目項目項目/SMS項目
項目 PLMN	項目項目項目	“10GB 項目項目”	項目項目項目項目/SMS項目

□ □ □ □ □ □ □ □

CGRateS 000000 ID 00000000000000000000 PLMN 000 0 **TarifPlan** 00
0000 0000000000000000000000000000

□□□□□

- [illegible]

1

□ □ □ □ □ □ □ □ □ □ □ □ □ □

```
# Initial configuration - Only 3 countries
OCS_Obj.SendData({
  'method': 'ApierV2.SetTPDestination',
  'params': [{
    "TPid": tpid,
    "ID": "Dest_PLMN_Zone_Europe",
    "Prefixes": [
      "mcc262", # Germany
      "mcc208", # France
      "mcc222"  # Italy
    ]
  }]
})
```

□□□□□□ □□ Dest_PLMN_Zone_Europe □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□


```

# Remove Germany from Europe zone
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_Europe",
        "Prefixes": [
            "mcc208", # France
            "mcc222", # Italy
            "mcc214", # Spain
            "mcc234"  # UK
            # Germany (mcc262) REMOVED
        ]
    }]
})

# Load the tariff plan to make changes active
OCS_Obj.SendData({
    'method': 'ApierV1.LoadTariffPlanFromStorDb',
    'params': [{
        "TPid": tpid,
        "DryRun": False,
        "Validate": True
    }]
})

# Reload cache
OCS_Obj.SendData({
    'method': 'CacheSv1.ReloadCache',
    'params': [{
        "DestinationIDs": ["Dest_PLMN_Zone_Europe"]
    }]
})

```

0000 0000000000000000000000000000 0000000000000000 Dest_PLMN_Zone_Europe 00000000
 00000 PLMN 00000

00 30000/SMS 000000

000000000000 SMS 00000000

DestinationIDs: "*any"

3#####

```
# Customer has:
# Balance 1: UK minutes
{
  "BalanceId": "UK_Voice_100min",
  "BalanceType": "*voice",
  "DestinationIDs": "Dest_International_UK",
  "Units": 100 * 60 * 1000000000,
  "BalanceWeight": 1200
}

# Balance 2: Monetary with wildcard
{
  "BalanceId": "PAYG_Credit",
  "BalanceType": "*monetary",
  "DestinationIDs": "*any", # Matches ANYTHING
  "Units": 5000, # $50
  "BalanceWeight": 1000 # Lower priority
}

# Customer calls Australia (+61)
```

###

- 61
- PAYG "*any"
- PAYG

#####

#####

- PLMN
- /
-

- □□□□□□□□

[illegible]

--	--	--	--

1. 00000000 - 00000000000000000000000000000000
2. 00 **TarifPlan** 000 - 00 LoadTariffPlanFromStorDb 0000000000000000
3. 00000000 = 0000 - 00000000000000000000000000000000
4. 0000 *any 000000 - 000000000000000000
5. 000000000 - 000000000000000000000000
6. 00000000 - 000000000000000000000000

PAYG/

PAYG - 00000000000000000000000000000000

CGRateS CGRateS

- 國際行動電話PLMN - 國際行動電話
- 國際行動電話網絡代碼 - 國際行動電話
- 國際行動電話網絡/國際行動電話
- 國際行動電話VIP國際行動電話
- 國際行動電話
- 國際行動電話網絡代碼

PLMN CRM
CGRateS CRM


```

#
=====
# VOICE RATES (Per Minute)
#
=====

# Domestic voice: $0.10/minute
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPRate',
    'params': [{
        "TPid": tpid,
        "ID": "Rate_Voice_Domestic",
        "RateSlots": [{
            "ConnectFee": 0,
            "Rate": 0.10,
            "RateUnit": "60s",          # 1 minute
            "RateIncrement": "1s",      # Per-second billing
            "GroupIntervalStart": "0s"
        }]
    }]
})

# UK voice: $0.25/minute
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPRate',
    'params': [{
        "TPid": tpid,
        "ID": "Rate_Voice_UK",
        "RateSlots": [{
            "ConnectFee": 0.05,          # $0.05 connection fee
            "Rate": 0.25,
            "RateUnit": "60s",
            "RateIncrement": "6s",      # 6-second blocks
            "GroupIntervalStart": "0s"
        }]
    }]
})

#
=====
# SMS RATES (Per Message)
#
=====

```

```

# Domestic SMS: $0.05/message
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPRate',
    'params': [{
        "TPid": tpid,
        "ID": "Rate_SMS_Domestic",
        "RateSlots": [{
            "ConnectFee": 0,
            "Rate": 0.05,
            "RateUnit": "1",          # 1 SMS
            "RateIncrement": "1",     # Per message
            "GroupIntervalStart": "0s"
        }]
    }]
})

#
=====
# DATA RATES (Per MB) - For Roaming
#
=====

# Roaming on US Premium Networks (Verizon, AT&T): $2.00/MB
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPRate',
    'params': [{
        "TPid": tpid,
        "ID": "Rate_Data_Roaming_US_Premium",
        "RateSlots": [{
            "ConnectFee": 0,
            "Rate": 2.00,
            "RateUnit": "1048576",   # 1 MB in bytes
            "RateIncrement": "1024", # Per KB
            "GroupIntervalStart": "0s"
        }]
    }]
})

```

□□□□□□

- **ConnectFee** - □□□□□□□□□□
- **Rate** - □□ RateUnit □□□

- **RateUnit** - 60s 1MB 1
- **RateIncrement** -
- **GroupIntervalStart** -

DestinationRates

Link Domestic destination to Domestic voice rate

```
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestinationRate',
    'params': [{
        "TPid": tpid,
        "ID": "DR_Voice_Domestic",
        "DestinationRates": [{
            "DestinationId": "Dest_Domestic_All",
            "RateId": "Rate_Voice_Domestic",
            "RoundingMethod": "*up",
            "RoundingDecimals": 4
        }]
    }]
})
```

Link UK destination to UK voice rate

```
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestinationRate',
    'params': [{
        "TPid": tpid,
        "ID": "DR_Voice_UK",
        "DestinationRates": [{
            "DestinationId": "Dest_International_UK",
            "RateId": "Rate_Voice_UK",
            "RoundingMethod": "*up",
            "RoundingDecimals": 4
        }]
    }]
})
```

Link Verizon PLMN to roaming data rate

```
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestinationRate',
    'params': [{
        "TPid": tpid,
        "ID": "DR_Data_Roaming_US_Premium",
        "DestinationRates": [{
            "DestinationId": "Dest_PLMN_US_Verizon",
            "RateId": "Rate_Data_Roaming_US_Premium",
            "RoundingMethod": "*up",
            "RoundingDecimals": 4
        }]
    }]
})
```

```
}]
})
```

□□□□□□

□□□ DestinationRates □□□□□□□□□□

```
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPRatingPlan',
    'params': [{
        "TPid": tpid,
        "ID": "RatingPlan_Standard_PAYG",
        "RatingPlanBindings": [
            # Voice rates (higher weight = more specific)
            {"DestinationRatesId": "DR_Voice_UK", "TimingId":
"*any", "Weight": 40},
            {"DestinationRatesId": "DR_Voice_Domestic",
"TimingId": "*any", "Weight": 20},

            # SMS rates
            {"DestinationRatesId": "DR_SMS_Domestic", "TimingId":
"*any", "Weight": 20},

            # Data roaming rates
            {"DestinationRatesId": "DR_Data_Roaming_US_Premium",
"TimingId": "*any", "Weight": 50}
        ]
    }]
})
```

□□□□□□

□□□□□□□□□□ CGRateS □□

```

# Load tariff plan
OCS_Obj.SendData({
    'method': 'ApierV1.LoadTariffPlanFromStorDb',
    'params': [{
        "TPid": tpid,
        "DryRun": False,
        "Validate": True
    }]
})

# Reload cache
OCS_Obj.SendData({
    'method': 'CacheSv1.ReloadCache',
    'params': [{
        "Tenant": tenant,
        "DestinationIDs": ["*all"]
    }]
})

```

□□□□□

- □□□□ - □□□□□□□□□□□□
- CGRateS □□□□□□□□ - □□□□□□□□□□□□

□□□□□

1. □□□□□□□□□□

```

# Good - specific home network
"DestinationIDs": "Dest_PLMN_OnNet", # mcc505.mnc057

# Bad - too broad, matches competitors
"DestinationIDs": "Dest_PLMN_AU_All", # mcc505 (all AU operators)

```

2. 命名空间命名

```
# Good - self-documenting
Dest_PLMN_US_Verizon
Dest_International_UK
Dest_PLMN_Zone_Europe

# Bad - unclear
Dest_Data_1
Dest_Voice_Package
```

3. 命名空间命名

```
# Good - logical grouping
Dest_Domestic_All
Dest_Domestic_TollFree
Dest_Domestic_Premium

# Bad - too granular
Dest_Area_Code_212
Dest_Area_Code_213
Dest_Area_Code_214
```

4. 命名 PLMN 命名

命名 PLMN 命名空间命名 MNC/MNC 命名空间

```

# Verizon Wireless
# Source: https://www.mcc-mnc.com/ (verified 2024-12)
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_US_Verizon",
        "Prefixes": [
            "mcc310.mnc004", # Verizon - Nationwide
            "mcc310.mnc010", # Verizon - East
            ...
        ]
    }]
})

```

□□□□

□□□□□□□□□□

□□□ □□□□□□□□□□□□□□□□

□□□

```

# 1. □□□□□□□□□□
OCS_Obj.SendData({
    'method': 'ApierV2.GetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_US_Verizon"
    }]
})

```

```

# 2. □□□□ DestinationIDs □□
# □□□□□□□□□□ "*any"

```

```

# 3. □□ PLMN□□□□□□□□□□□□□□□□
# □□ Diameter CCR □□□□ Visited-PLMN-Id AVP

```

PLMN 設定

PLMN 設定の概要

PLMN 設定の PLMN ID は `Dest_PLMN_OnNet` である

設定例

```
# PLMN 設定 Diameter CCR  
# PLMN 設定 PLMN  
  
OCS_Obj.SendData({  
    'method': 'ApierV2.GetTPDestination',  
    'params': [{  
        "TPid": tpid,  
        "ID": "Dest_PLMN_OnNet"  
    }]  
})  
# PLMN 設定 PLMN  
# ["mcc505.mnc057"] SIM 設定
```

PLMN 設定

PLMN 設定の PLMN ID

設定

- PLMN 設定の PLMN ID
◦ `+44-20-xxxx` → `"44"`
- PLMN **DestinationIDs** 設定
◦ PLMN 設定 `"DestinationIDs": "Dest_International_UK"` 又は `"Dest_International_All"`
- PLMN 設定 `ExpiryTime`
- PLMN 設定 `Units > 0`

PLMN 設定

- CGRateS** 設定 - PLMN 設定

- 0000 - 000000000000

Ansible Playbooks: Overview

OmniCRM uses **Ansible** to manage and provision SIM cards.

Concepts SIM Card Provisioning <concepts_sim_provisioning> Ansible
SIM and eSIM

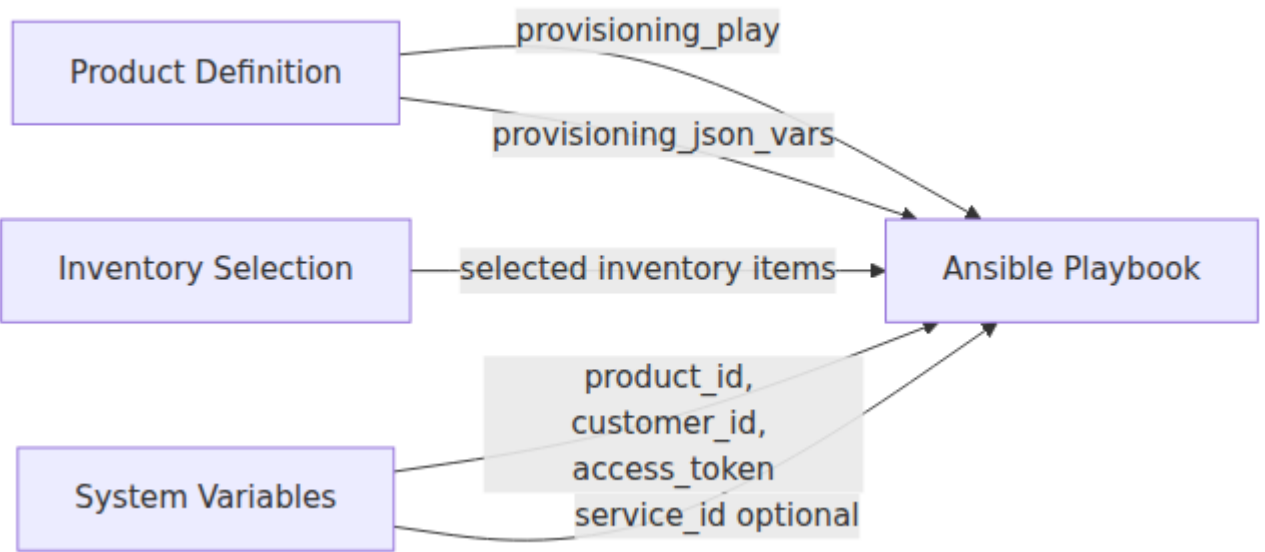
Playbooks Overview

Playbooks in OmniCRM are Ansible playbooks used to manage SIM cards. - Detailed description of the playbooks.

Ansible Playbooks

OmniCRM uses Ansible

1. provisioning_play
2. provisioning_json_vars
3. playbook
4. playbook



Playbooks 目錄

目錄 playbook 目錄

目錄

目錄 playbook 目錄

- 目錄
- 目錄 IPTV 目錄
- 目錄 VoIP 目錄
- 目錄

目錄

目錄 playbooks 目錄

- 目錄 CPE 目錄 playbook
- 目錄 playbook
- 目錄 playbook

目錄

目錄

- 目錄
- 目錄
- 目錄

目錄

目錄 playbooks

- 目錄
- 目錄
- 目錄

目錄

目錄 playbooks 目錄

- 目錄
- 目錄

- □□□□□□

Playbook

```
# [] 1[] SIM [] [] 1 []
# provisioning_play: play_simple_service
- [] CRM []
- [] OCS []
- [] SIM []
- []

# [] 2[][] [] 3 []
# provisioning_play: play_bundle_internet_tv_voice
- []
- [] IPTV []
- [] VoIP []
- []
- []

# [] 3[] [] 0 []
# provisioning_play: play_topup_no_charge
- [] service_id []
- [] OCS []
- []
- []

# [] 4[CPE] [] [] 0 []
# provisioning_play: play_prov_cpe_mikrotik
- []
- []
- []
- []
```

playbook

Plays ☐ Tasks

Plays Tasks OmniCRM playbooks

Playbook

Playbooks are located in the `OmniCRM-API/Provisioners/plays/` directory.

There are three

- `play_simple_service.yaml` - Simple service
- `play_topup_no_charge.yaml` - Topup without charge
- `play_prov_cpe_mikrotik.yaml` - Provisioning CPE Mikrotik

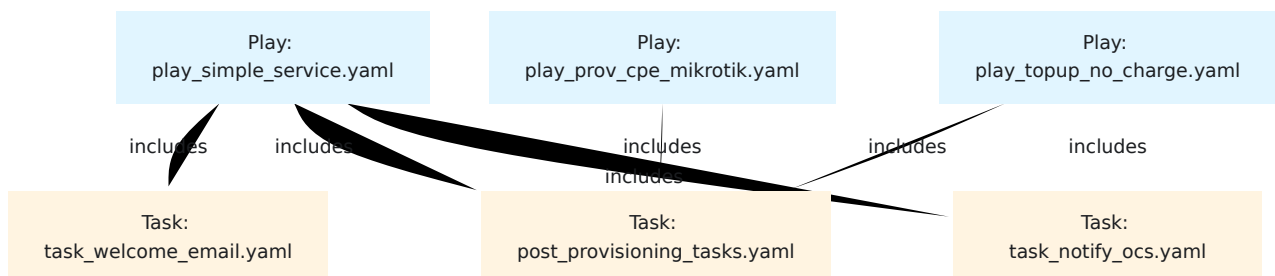
Task

Tasks are located in the `task_` directory.

There are three

- `task_welcome_email.yaml` - Welcome email
- `task_activate_olt.yaml` - Activate OLT
- `task_notify_ocs.yaml` - Notify OCS

Diagram



```
# play_simple_service.yaml (A Play)
- name: Simple Provisioning Play
  hosts: localhost
  tasks:
    - name: Main provisioning block
      block:
        - name: Create service
          uri: ...

        - name: Configure billing
          uri: ...

    # Include reusable task
    - include_tasks: task_welcome_email.yaml

    # Include post-provisioning tasks
    - include_tasks: post_provisioning_tasks.yaml
```

Playbook

OmniCRM playbooks playbooks

□ □ □ □

playbook

```
- name: Descriptive Name of the Playbook
  hosts: localhost # [] localhost
  gather_facts: no # []
  become: False # []

tasks:
  - name: Main block
    block:
      # []

    rescue:
      # []/[]
```

#####

name

UI ##### `playbook_description`

hosts: localhost

OmniCRM playbooks ## localhost ##### API ##### SSH

gather_facts: no

Ansible #####

- #####
- #####
- #####

become: False

API

#####

playbook

```
tasks:
  - name: Include vars of crm_config
    ansible.builtin.include_vars:
      file: "../..//crm_config.yaml"
      name: crm_config
```

`crm_config.ocs.cgrates` `crm_config.crm.base_url`

`crm_config.yaml` #####

```
ocs:
  cgrates: "10.0.1.100:2080"
  ocsTenant: "default_tenant"
crm:
  base_url: "https://crm.example.com"
```

#####

#####

#####

```
- name: Access product_id passed by OmniCRM
  debug:
    msg: "Provisioning product {{ product_id }}"
```

#####

```
- name: Get inventory ID for SIM Card
  set_fact:
    sim_card_id: "{{ hostvars[inventory_hostname]['SIM Card'] |
int }}"
  when: "'SIM Card' in hostvars[inventory_hostname]"
```

API

```
- name: Get Product information from CRM API
  uri:
    url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
    return_content: yes
  register: api_response_product

- name: Use the product name
  debug:
    msg: "Product name is {{
api_response_product.json.product_name }}"
```

📖 **Playbook** 📖

📌📌📌📌📌

📌📌📌📌📌📌📌📌📌📌

```

- name: Service Provisioning Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Main block
      block:

        # 1. Include
        - name: Include vars of crm_config
          ansible.builtin.include_vars:
            file: "../..crm_config.yaml"
            name: crm_config

        # 2. Get Product Information
        - name: Get Product information from CRM API
          uri:
            url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
            method: GET
            headers:
              Authorization: "Bearer {{ access_token }}"
            return_content: yes
            validate_certs: no
            register: api_response_product

        # 3. Get Customer Information
        - name: Get Customer information from CRM API
          uri:
            url: "http://localhost:5000/crm/customer/customer_id/{{
customer_id }}"
            method: GET
            headers:
              Authorization: "Bearer {{ access_token }}"
            return_content: yes
            register: api_response_customer

        # 4. Set Package Facts
        - name: Set package facts
          set_fact:
            package_name: "{{ api_response_product.json.product_name
}}"

```



```
package_comment: "{{ api_response_product.json.comment
}}"
setup_cost: "{{
api_response_product.json.retail_setup_cost }}"
monthly_cost: "{{ api_response_product.json.retail_cost
}}"
```

5. 生成UUID

- name: Generate UUID
set_fact:
 uuid: "{{ 99999999 | random | to_uuid }}"
- name: Generate Service UUID
set_fact:
 service_uuid: "Service_{{ uuid[0:8] }}"

6. 创建OCS账户

- name: Create account in OCS/CGRateS
uri:
 url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
 method: POST
 body_format: json
 headers:
 Content-Type: "application/json"
 body:
 {
 "method": "ApierV2.SetAccount",
 "params": [{
 "Tenant": "{{ crm_config.ocs.ocsTenant }}",
 "Account": "{{ service_uuid }}",
 "ActionPlanIds": [],
 "ActionPlansOverwrite": true,
 "ExtraOptions": {
 "AllowNegative": false,
 "Disabled": false
 },
 "ReloadScheduler": true
 }]
 }
 status_code: 200
 register: ocs_response
- name: Verify OCS account creation
assert:

```

    that:
      - ocs_response.status == 200
      - ocs_response.json.result == "OK"

# 7. 余额管理
- name: Add 0 Monetary Balance
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "BalanceType": "*monetary",
          "Categories": "*any",
          "Balance": {
            "ID": "Initial Balance",
            "Value": 0,
            "ExpiryTime": "+4320h",
            "Weight": 1,
            "Blocker": true
          }
        }]
      }
    status_code: 200
  register: balance_response

# 8. CRM 系统管理
- name: Get current date and time in ISO 8601 format
  command: date --utc +%Y-%m-%dT%H:%M:%S%z
  register: current_date_time

- name: Add Service via API
  uri:
    url: "http://localhost:5000/crm/service/"
    method: PUT
    body_format: json
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body:

```

```

    {
      "customer_id": "{{ customer_id }}",
      "product_id": "{{ product_id }}",
      "service_name": "{{ package_name }} - {{
service_uuid }}",
      "service_type": "generic",
      "service_uuid": "{{ service_uuid }}",
      "service_billed": true,
      "service_taxable": true,
      "service_provisioned_date": "{{
current_date_time.stdout }}",
      "service_status": "Active",
      "wholesale_cost": "{{
api_response_product.json.wholesale_cost | float }}",
      "retail_cost": "{{ monthly_cost | float }}"
    }
    status_code: 200
    register: service_creation_response

```

9. 〇〇〇〇〇〇〇〇

```

- name: Add Setup Cost Transaction via API
  uri:
    url: "http://localhost:5000/crm/transaction/"
    method: PUT
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": {{ customer_id | int }},
        "service_id": {{
service_creation_response.json.service_id | int }},
        "title": "{{ package_name }} - Setup Costs",
        "description": "Setup costs for {{ package_comment
}}",
        "invoice_id": null,
        "retail_cost": "{{ setup_cost | float }}"
      }
    return_content: yes
    register: transaction_response

```

10. 〇〇〇〇〇〇〇〇

```

- include_tasks: post_provisioning_tasks.yaml

```

```

rescue:

# 00/0000
- name: Print all vars for debugging
  debug:
    var: hostvars[inventory_hostname]

- name: Remove account in OCS
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV2.RemoveAccount",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "ReloadScheduler": true
        }]
      }
    status_code: 200
    ignore_errors: True
    when: service_uuid is defined

- name: Delete Service from CRM if it was created
  uri:
    url: "http://localhost:5000/crm/service/service_id/{{
service_creation_response.json.service_id }}"
    method: DELETE
    headers:
      Authorization: "Bearer {{ access_token }}"
    status_code: 200
    ignore_errors: True
    when: service_creation_response is defined

- name: Fail if not intentional deprovision
  assert:
    that:
      - action == "deprovision"

```

Topup/Recharge ☐

```

- name: Service Topup Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Include vars of crm_config
      ansible.builtin.include_vars:
        file: "../..//crm_config.yaml"
        name: crm_config

    # 1. 获取服务信息
    - name: Get Service information from CRM API
      uri:
        url: "http://localhost:5000/crm/service/service_id/{{
service_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_service

    # 2. 获取产品信息
    - name: Get Product information from CRM API
      uri:
        url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_product

    # 3. 设置服务事实
    - name: Set service facts
      set_fact:
        service_uuid: "{{ api_response_service.json.service_uuid
}}"

        customer_id: "{{ api_response_service.json.customer_id }}"
        package_name: "{{ api_response_product.json.product_name
}}"

        topup_value: "{{ api_response_product.json.retail_cost }}"

```

4. 测试添加信用

```
- name: Execute Action to add credits
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "APIerSv1.ExecuteAction",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "ActionsId": "Action_Topup_Standard"
      }]
    }
  status_code: 200
register: action_response
```

```
- name: Verify action executed successfully
assert:
  that:
    - action_response.status == 200
    - action_response.json.result == "OK"
```

5. 测试重置

```
- name: Reset ActionTriggers
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "APIerSv1.ResetAccountActionTriggers",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "Executed": false
      }]
    }
  status_code: 200
```

6. 测试过期

```
- name: Calculate new expiry date
command: "date --utc +%Y-%m-%dT%H:%M:%S%z -d '+30 days'"
```

```

    register: new_expiry_date

- name: Update Service with new expiry
  uri:
    url: "http://localhost:5000/crm/service/{{ service_id }}"
    method: PATCH
    headers:
      Authorization: "Bearer {{ access_token }}"
      Content-Type: "application/json"
    body_format: json
    body:
      {
        "service_deactivate_date": "{{ new_expiry_date.stdout
      }},
        "service_status": "Active"
      }

# 7. 〇〇〇〇〇〇〇〇
- name: Send Notification SMS
  uri:
    url: "http://sms-gateway/api/send"
    method: POST
    body_format: json
    body:
      {
        "source": "CompanyName",
        "destination": "{{ customer_phone }}",
        "message": "Your service has been topped up. New
expiry: {{ new_expiry_date.stdout }}"
      }
    status_code: 201
    ignore_errors: True

```

CPE 〇〇〇〇

〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇ONT〇〇


```

- name: CPE Provisioning Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Include vars of crm_config
      ansible.builtin.include_vars:
        file: "../crm_config.yaml"
        name: crm_config

    # 1. 设置 CPE 库存 ID
    - name: Set CPE inventory ID from hostvars
      set_fact:
        cpe_inventory_id: "{{ hostvars[inventory_hostname]['WiFi Router CPE'] | int }}"
        when: "'WiFi Router CPE' in hostvars[inventory_hostname]"

    # 2. 获取 CPE 库存数据
    - name: Get Inventory data for CPE
      uri:
        url: "{{ crm_config.crm.base_url
        }}/crm/inventory/inventory_id/{{ cpe_inventory_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_cpe

    # 3. 获取站点信息
    - name: Get Site info from API
      uri:
        url: "{{ crm_config.crm.base_url
        }}/crm/site/customer_id/{{ customer_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_site

    # 4. 更新 CPE 位置
    - name: Patch CPE inventory item with location
      uri:

```

```

        url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ cpe_inventory_id }}"
        method: PATCH
        body_format: json
        headers:
            Authorization: "Bearer {{ access_token }}"
        body:
            {
                "address_line_1": "{{
api_response_site.json.0.address_line_1 }}",
                "city": "{{ api_response_site.json.0.city }}",
                "state": "{{ api_response_site.json.0.state }}",
                "latitude": "{{ api_response_site.json.0.latitude }}",
                "longitude": "{{ api_response_site.json.0.longitude
}}"
            }
        status_code: 200

```

5. 配置 CPE

```

- name: Set CPE hostname
  set_fact:
    cpe_hostname: "CPE-{{ cpe_inventory_id }}"
    cpe_username: "admin-{{ cpe_inventory_id }}"

- name: Generate random password
  set_fact:
    cpe_password: "{{ lookup('pipe', 'cat /dev/urandom | tr -
dc a-zA-Z0-9 | head -c 16') }}"

```

6. 配置 WiFi 网络

```

- name: Set WiFi SSID
  set_fact:
    wifi_ssid: "Network-{{ cpe_inventory_id }}"

- name: Generate WiFi password
  set_fact:
    word_list:
      - apple
      - cloud
      - river
      - mountain
      - ocean

- name: Create WiFi PSK

```

```

set_fact:
    random_word: "{{ word_list | random }}"
    random_number: "{{ 99999 | random(start=10000) }}"

- name: Combine WiFi PSK
  set_fact:
    wifi_psk: "{{ random_word }}{{ random_number }}"

# 7. 配置路由器
- name: Set config filename
  set_fact:
    config_name: "{{ cpe_hostname }}_{{ lookup('pipe', 'date
+%Y%m%d%H%M%S') }}.cfg"
    config_dest: "/tmp/{{ cpe_hostname }}_{{ lookup('pipe',
'date +%Y%m%d%H%M%S') }}.cfg"

- name: Create config from template
  template:
    src: "templates/cpe_router_config.j2"
    dest: "{{ config_dest }}"

# 8. 读取配置文件
- name: Read config file
  ansible.builtin.slurp:
    src: "{{ config_dest }}"
  register: config_content

# 9. 打补丁
- name: Patch CPE inventory with config
  uri:
    url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ cpe_inventory_id }}"
    method: PATCH
    body_format: json
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      {
        "itemtext3": "{{ wifi_ssid }}",
        "itemtext4": "{{ wifi_psk }}",
        "management_url": "{{ cpe_hostname }}",
        "management_username": "{{ cpe_username }}",
        "management_password": "{{ cpe_password }}",
        "config_content": "{{ config_content.content |

```

```

b64decode }}",
    "inventory_notes": "Provisioned: {{ lookup('pipe',
'date +%Y-%m-%d') }}"
    }
    status_code: 200

# 10. 配置邮件
- name: Email configuration to support
  uri:
    url: "https://api.mailjet.com/v3.1/send"
    method: POST
    body_format: json
    headers:
      Content-Type: "application/json"
    body:
      {
        "Messages": [{
          "From": {
            "Email": "provisioning@example.com",
            "Name": "Provisioning System"
          },
          "To": [{
            "Email": "support@example.com",
            "Name": "Support Team"
          }],
          "Subject": "CPE Config - {{ cpe_hostname }}",
          "Attachments": [{
            "ContentType": "text/plain",
            "Filename": "{{ config_name }}",
            "Base64Content": "{{ config_content.content }}"
          }]
        }]
      }
    user: "{{ mailjet_api_key }}"
    password: "{{ mailjet_api_secret }}"
    force_basic_auth: true
    status_code: 200

```

配置邮件

配置 CGRateS ActionPlans 配置邮件

```

# 创建 playbook 目录

# 1. 创建 auto_renew 目录
- name: Normalize auto_renew to boolean
  set_fact:
    auto_renew_bool: "{{ (auto_renew | string | lower) in ['true', '1', 'yes'] }}"

# 2. 创建 Action 目录
- name: Create Action for AutoRenew
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.SetActions",
        "params": [{
          "ActionsId": "Action_AutoTopup_{{ service_uuid }}_{{
product_id }}",
          "Overwrite": true,
          "Actions": [
            {
              "Identifier": "*http_post",
              "ExtraParameters": "{{ crm_config.crm.base_url
}}/crm/provision/simple_provision_addon/service_id/{{ service_id
}}/product_id/{{ product_id }}"
            },
            {
              "Identifier": "*cdrlog",
              "BalanceType": "*generic",
              "ExtraParameters": "
{{\"Category\\\": \"^activation\\\", \"Destination\\\": \"Auto Renewal\\\"}}"
            }
          ]
        }]
      }
    status_code: 200
    register: action_response
    when: auto_renew_bool

# 3. 创建 ActionPlan
- name: Create ActionPlan for Monthly Renewal

```

```

uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.SetActionPlan",
      "params": [{
        "Id": "ActionPlan_Monthly_{{ service_uuid }}_{{
product_id }}" ,
        "Tenant": "{{ crm_config.ocs.ocsTenant }}" ,
        "ActionPlan": [{
          "ActionsId": "Action_AutoTopup_{{ service_uuid }}_{{
product_id }}" ,
          "Years": "*any",
          "Months": "*any",
          "MonthDays": "*any",
          "WeekDays": "*any",
          "Time": "*monthly",
          "StartTime": "*now",
          "Weight": 10
        }],
        "Overwrite": true,
        "ReloadScheduler": true
      }]
    }
  status_code: 200
when: auto_renew_bool

```

4. `ActionPlan` 配置

- name: Assign ActionPlan to account

```

uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV2.SetAccount",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}" ,
        "Account": "{{ service_uuid }}" ,
        "ActionPlanIds": ["ActionPlan_Monthly_{{ service_uuid
}}_{{ product_id }}"],
        "ActionPlansOverwrite": true,

```

```

        "ReloadScheduler": true
    }
}
status_code: 200
when: auto_renew_bool

# 5. 删除行动计划 ActionPlan
- name: Remove ActionPlan from account
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.RemoveActionPlan",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Id": "ActionPlan_Monthly_{{ service_uuid }}_{{
product_id }}"
        }]
      }
    status_code: 200
    ignore_errors: true
    when: not auto_renew_bool

```

任务清单

任务清单 playbook 任务清单 plays 任务清单

任务清单

task_welcome_email.yaml - 任务清单

```

# [] [] [] [] [] [] [] [] play [] []
# - api_response_customer ( [] [] [] [] [] )
# - package_name ( [] [] [] [] )
# - monthly_cost ( [] [] [] [] )
# - setup_cost ( [] [] [] [] )

- name: Set email configuration
  set_fact:
    mailjet_api_key: "{{ lookup('env', 'MAILJET_API_KEY') }}"
    mailjet_api_secret: "{{ lookup('env', 'MAILJET_SECRET') }}"
    email_from: "noreply@example.com"
    recipients: []

- name: Set email subject and sender name
  set_fact:
    email_subject: "Welcome to our service!"
    email_from_name: "Customer Service Team"

- name: Prepare list of recipients from customer contacts
  loop: "{{ api_response_customer.json.contacts }}"
  set_fact:
    recipients: "{{ recipients + [{'Email': item.contact_email,
'Name': item.contact_firstname ~ ' ' ~ item.contact_lastname}] }}"

- name: Get first contact name
  set_fact:
    first_contact: "{{
api_response_customer.json.contacts[0].contact_firstname }}"

- name: Send welcome email
  uri:
    url: "https://api.mailjet.com/v3.1/send"
    method: POST
    body_format: json
    headers:
      Content-Type: "application/json"
    body:
      {
        "Messages": [{
          "From": {
            "Email": "{{ email_from }}",
            "Name": "{{ email_from_name }}"
          },

```



```

        "To": "{{ recipients }}",
        "Subject": "{{ email_subject }}",
        "TextPart": "Dear {{ first_contact }}, welcome! Your
service is ready.",
        "HTMLPart": "Dear {{ first_contact }},<br/><h3>Welcome!
</h3><br/>Your {{ package_name }} service is now active.
<br/>Monthly cost: ${{ monthly_cost }}<br/>Setup fee: ${{
setup_cost }}<br/>If you have any questions, contact
support@example.com"
    }
}
}
user: "{{ mailjet_api_key }}"
password: "{{ mailjet_api_secret }}"
force_basic_auth: true
status_code: 200
register: email_response

```

□□□□□

post_provisioning_tasks.yaml - □□□□□□□□□□□□□□□□

```

# □□□□□□□□□□ playbooks □□□□□
# □□□□□□□□□□□□

- include_tasks: task_notify_ocs.yaml

```

task_notify_ocs.yaml □□□□□

```
- name: Notify OCS of provisioning completion
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "APIerSv1.ReloadCache",
        "params": [{
          "ArgsCache": "*all"
        }]
      }
    status_code: 200
    ignore_errors: true
```

□□□□

□□□□□

□□□□□□□□

```

- name: Get SIM Card inventory ID
  set_fact:
    sim_inventory_id: "{{ hostvars[inventory_hostname]['SIM Card']
| int }}"
  when: "'SIM Card' in hostvars[inventory_hostname]"

- name: Get SIM Card details
  uri:
    url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ sim_inventory_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
    return_content: yes
    register: sim_response

- name: Extract SIM details
  set_fact:
    iccid: "{{ sim_response.json.iccid }}"
    imsi: "{{ sim_response.json.imsi }}"
    ki: "{{ sim_response.json.ki }}"

```

□□□□□□□□□□

```

- name: Assign SIM to customer
  uri:
    url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ sim_inventory_id }}"
    method: PATCH
    headers:
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": {{ customer_id }},
        "service_id": {{ service_id }},
        "item_state": "Assigned"
      }
    status_code: 200

```

日期时间

日期时间/时间

- **name**: Get current date and time in ISO 8601 format
command: `date --utc +%Y-%m-%dT%H:%M:%S%z`
register: `current_date_time`
- **name**: Get today's date only
set_fact:
 today: `"{{ lookup('pipe', 'date +%Y-%m-%d') }}"`

日期

- **name**: Calculate expiry date 30 days from now
command: `"date --utc +%Y-%m-%dT%H:%M:%S%z -d '+30 days'"`
register: `expiry_date`
- **name**: Calculate date 90 days in future
command: `"date --utc +%Y-%m-%d -d '+{{ days }} days'"`
register: `future_date`
vars:
 days: 90

UUID

UUID 生成

- **name**: Generate UUID
set_fact:
 uuid: `"{{ 99999999 | random | to_uuid }}"`
- **name**: Generate service identifier
set_fact:
 service_uuid: `"SVC_{{ uuid[0:8] }}"`

服务标识

```
- name: Generate secure password
  set_fact:
    password: "{{ lookup('pipe', 'cat /dev/urandom | tr -dc a-zA-Z0-9 | head -c 16') }}"
```

□□□□□□□□

```
- name: Set word list
  set_fact:
    words:
      - alpha
      - bravo
      - charlie
      - delta
      - echo

- name: Generate passphrase
  set_fact:
    word: "{{ words | random }}"
    number: "{{ 99999 | random(start=10000) }}"

- name: Combine into passphrase
  set_fact:
    passphrase: "{{ word }}{{ number }}"
```

□ **CGRateS/OCS** □□

□□□□□

```
- name: Create billing account
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV2.SetAccount",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "ActionPlanIds": [],
        "ActionPlansOverwrite": true,
        "ExtraOptions": {
          "AllowNegative": false,
          "Disabled": false
        },
        "ReloadScheduler": true
      }]
    }
  status_code: 200
register: account_response
```

□□□□□

```
- name: Add data balance
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.AddBalance",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "BalanceType": "*data",
        "Categories": "*any",
        "Balance": {
          "ID": "Data Package",
          "Value": 10737418240,
          "ExpiryTime": "+720h",
          "Weight": 10
        }
      }]
    }
  status_code: 200
```

□□□□□

```
- name: Execute charging action
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "APIerSv1.ExecuteAction",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "ActionsId": "Action_Standard_Charge"
      }]
    }
  status_code: 200
```

□□□□□□□□

```
- name: Get account details
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV2.GetAccount",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}"
        }]
      }
    status_code: 200
  register: account_info
```

□□□□□□□□□□


```

- name: Get AttributeProfile
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "APIerSv1.GetAttributeProfile",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "ID": "ATTR_{{ service_uuid }}"
        }]
      }
    return_content: yes
    status_code: 200
  register: attr_response
  ignore_errors: true

- name: Extract attribute value
  set_fact:
    phone_number: "{{ attr_response.json.result.Attributes |
  json_query('\[?Path=='*req.PhoneNumber'].Value[0].Rules\') | first
  }}"
  when: attr_response is defined

```

□□□□

□□□□□□□□

```

- name: Use custom value or default
  set_fact:
    monthly_cost: "{{ custom_cost | default(50.00) }}"

- name: Only run if variable is defined
  debug:
    msg: "Service UUID is {{ service_uuid }}"
  when: service_uuid is defined

```

□□□□

- `name`: Provision equipment
`include_tasks`: `configure_cpe.yaml`
`when`: `provision_cpe | default(false) | bool`
- `name`: Skip if deprovision
`assert`:
 - `that`:
 - `action != "deprovision"`
 - `when`: `action is defined`

□□□□□

- `name`: Complex conditional task
`uri`:
 - `url`: `"{{ endpoint }}"`
 - `method`: `POST``when`:
 - `service_uuid is defined`
 - `customer_id is defined`
 - `action != "deprovision"`
 - `enable_feature | default(true) | bool`

□□□□□

□□□□□

```

- name: Create multiple balances
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Account": "{{ service_uuid }}",
          "BalanceType": "{{ item.type }}",
          "Balance": {
            "Value": "{{ item.value }}"
          }
        }]
      }
  loop:
    - { type: "*voice", value: 3600 }
    - { type: "*data", value: 10737418240 }
    - { type: "*sms", value: 100 }

```

API

```

- name: Get all customer sites
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/site/customer_id/{{ customer_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: sites_response

- name: Configure equipment at each site
  debug:
    msg: "Configuring site at {{ item.address_line_1 }}"
  loop: "{{ sites_response.json }}"

```

ignore_errors

```
- name: Optional SMS notification
  uri:
    url: "http://sms-gateway/send"
    method: POST
    body: {...}
  ignore_errors: true
```

□□□□□□□□

```
- name: Verify API response
  assert:
    that:
      - response.status == 200
      - response.json.result == "OK"
  fail_msg: "API call failed: {{ response.json }}"
```

□□□□□□□□

```
- name: Try to get existing service
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/service/service_uuid/{{ service_uuid }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: service_lookup
  failed_when: false

- name: Create service if it doesn't exist
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/service/"
    method: PUT
    body: {...}
  when: service_lookup.status == 404
```

□□□□

□□□□

□□□□□□□□□□

```
# □
service_uuid: "SVC_12345"
customer_name: "John Smith"
monthly_cost: 49.99
```

```
# □
svc: "SVC_12345"
name: "John Smith"
cost: 49.99
```

□□□□□□□□

```
api_response_customer: {...}
api_response_product: {...}
cgr_account_info: {...}
```

□□

□□□□□□□□□□

- name: Print all variables
debug:
var: hostvars[inventory_hostname]
- name: Print specific variable
debug:
msg: "Service UUID: {{ service_uuid }}"
- name: Print API response
debug:
var: api_response_product.json

□□

□□□□□□ API □□□

```
- name: Create account
  uri:
    url: "{{ billing_endpoint }}"
    method: POST
    body: {...}
  register: response

- name: Verify account creation
  assert:
    that:
      - response.status == 200
      - response.json.result == "OK"
  fail_msg: "Failed to create account: {{ response.json }}"
```

□□□

□□□□□□□□□□□□□□

```
# □□□□□□□□□□
- name: Check if account exists
  uri:
    url: "{{ ocs_endpoint }}/get_account"
    method: POST
    body: {"Account": "{{ service_uuid }}" }
  register: account_check
  failed_when: false

# □□□□□□□□□□
- name: Create account
  uri:
    url: "{{ ocs_endpoint }}/create_account"
    method: POST
    body: {...}
  when: account_check.status == 404
```

□□□

□□□□□□□□□□

```
# □  
mailjet_api_key: "abc123def456"  
  
# □ - □□□□□□  
mailjet_api_key: "{{ lookup('env', 'MAILJET_API_KEY') }}"  
  
# □ - □□□□□□  
mailjet_api_key: "{{ crm_config.email.api_key }}"
```

□□□□ HTTPS □□□□□□

```
- name: Call external API  
  uri:  
    url: "https://api.example.com/endpoint"  
    method: POST  
    headers:  
      Authorization: "Bearer {{ access_token }}"  
    validate_certs: yes
```

□□

□□□□□□□□

```
# 1. 15 days before the end of the month
# 2. 15 days before the end of the month
# 3. 50% of the monthly cost

- name: Calculate days until end of month
  command: "date -d 'last day of this month' +%d"
  register: days_in_month

- name: Get current day
  command: "date +%d"
  register: current_day

- name: Calculate pro-rata amount
  set_fact:
    days_remaining: "{{ (days_in_month.stdout | int) -
(current_day.stdout | int) }}"
    pro_rata_cost: "{{ (monthly_cost | float) * (days_remaining |
float) / (days_in_month.stdout | float) }}"
```

Playbooks

Tasks

1. Create a new playbook
2. Create a new task
3. Create a new task with API key
4. Create a new task with API key
5. Create a new task with `action: "deprovision"` and `register: deprovision`

playbook


```

- name: Test Service Provisioning
  hosts: localhost
  gather_facts: no

  tasks:
    - name: Verify required variables
      assert:
        that:
          - product_id is defined
          - customer_id is defined
          - access_token is defined
        fail_msg: "Missing required variables"

    - name: Test API connectivity
      uri:
        url: "http://localhost:5000/crm/health"
        method: GET
      register: health_check

    - name: Verify health check
      assert:
        that:
          - health_check.status == 200

```

□□□□

□□□□□□□

```

# □□ - □□□□□□
customer_id: "{{ customer_id }}"

# □□ - □□□□□□
customer_id: {{ customer_id | int }}

```

□□□□□□□□□□

Ansible 簡介

Ansible playbooks 是什麼

Ansible

OmniCRM 系統部署

Ansible

系統部署SIM IP 系統部署

Ansible

OmniCRM 系統部署

- product_id - 產品ID
- customer_id - 客戶ID
- service_id - 服務ID/類別
- access_token - API 認證 JWT

Ansible Playbooks

Ansible Playbooks 是 rescue 系統部署

- 系統部署
- 系統部署
- 系統部署
- 系統部署

Ansible Playbooks

系統部署

1. 系統部署

系統部署

2. 系統部署

playbook 是 action: "deprovision" 系統部署

❏❏❏❏❏

❏❏ OmniCRM playbooks ❏❏❏❏❏❏❏❏❏❏

```
- name: Service Provisioning Playbook
  hosts: localhost
  gather_facts: no
  become: False

tasks:
  - name: Main block
    block:
      # ❏❏❏❏❏❏❏❏❏
      - name: Create OCS account
        uri: ...

      - name: Create service record
        uri: ...
        register: service_creation_response

      - name: Add transaction
        uri: ...

    rescue:
      # ❏❏❏/❏❏❏❏❏❏❏❏❏
      - name: Print all vars for debugging
        debug:
          var: hostvars[inventory_hostname]

      # ❏❏❏❏❏❏❏❏❏❏❏❏❏
      - name: Remove account in OCS
        uri: ...

      - name: Delete Service from CRM
        uri: ...

      - name: Fail if not intentional deprovision
        assert:
          that:
            - action == "deprovision"
```

□□□□□□

□□ `play_simple_service.yaml` □□□□

```

rescue:
  # 1. 初期化
  - name: Print all vars for Deprovision/Rollback
    debug:
      var: hostvars[inventory_hostname]

  - name: Get today's date
    set_fact:
      today: "{{ lookup('pipe', 'date +%Y-%m-%d') }}"

  # 2. 初期化
  - name: Try to get Service information from CRM API to
    Deprovision
    uri:
      url: "http://localhost:5000/crm/service/service_id/{{
service_creation_response.json.service_id }}"
      method: GET
      headers:
        Authorization: "Bearer {{ access_token }}"
      return_content: yes
    register: api_response_service
    ignore_errors: True
    when: service_creation_response is defined and
service_creation_response.json is defined and
service_creation_response.json.service_id is defined

  - name: Print api_response_service
    debug:
      var: api_response_service
    when: api_response_service is defined

  # 3. 初期化 service_uuid
  - name: Set service_uuid facts for Deprovision
    set_fact:
      service_uuid: "{{ api_response_service.json.service_uuid }}"
    when:
      - service_uuid is not defined
      - api_response_service is defined
      - api_response_service.json is defined

  # 4. OCS 初期化
  - name: Remove account in OCS
    uri:

```

```

url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
method: POST
body_format: json
headers:
  Content-Type: "application/json"
  Authorization: "Bearer {{ access_token }}"
body:
  {
    "method": "ApierV2.RemoveAccount",
    "params": [{
      "Tenant": "{{ crm_config.ocs.ocsTenant }}",
      "Account": "{{ service_uuid }}",
      "ReloadScheduler": true
    }]
  }
status_code: 200
register: response
ignore_errors: True
when: service_uuid is defined

```

- name: Print response from OCS account removal

```

debug:
  var: response
when: response is defined

```

5. CRM 删除服务

- name: Delete Service from CRM if it was created

```

uri:
  url: "http://localhost:5000/crm/service/service_id/{{
service_creation_response.json.service_id }}"
  method: DELETE
  headers:
    Authorization: "Bearer {{ access_token }}"
  status_code: 200
register: delete_service_response
ignore_errors: True
when: service_creation_response is defined and
service_creation_response.json is defined and
service_creation_response.json.service_id is defined

```

- name: Print delete service response

```

debug:
  var: delete_service_response
when: delete_service_response is defined

```

```
# 6. 状態確認
- name: Set status to "Success" if Manual deprovision / Fail if
  failed provision
  assert:
    that:
      - action == "deprovision"
    fail_msg: "Provisioning failed and was rolled back"
    success_msg: "Service deprovisioned successfully"
```

確認

ignore_errors: True

確認 `ignore_errors: True` 確認

確認

確認 `when` 確認

`when: service_creation_response is defined`

確認

確認

1. 確認
2. 確認
3. 確認 OCS/確認
4. 確認

確認

確認

- 確認 (`action == "deprovision"`) → 確認
- 確認 → 確認

確認

OCS/CGRateS 確認


```

- name: Remove account in OCS
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV2.RemoveAccount",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "ReloadScheduler": true
        }]
      }
    status_code: 200
  ignore_errors: True
  when: service_uuid is defined

```

CRM 删除服务

```

- name: Delete Service from CRM
  uri:
    url: "http://localhost:5000/crm/service/service_id/{{
service_id }}"
    method: DELETE
    headers:
      Authorization: "Bearer {{ access_token }}"
    status_code: 200
  ignore_errors: True
  when: service_id is defined

```

删除服务

```

- name: Release SIM card back to inventory pool
  uri:
    url: "http://localhost:5000/crm/inventory/inventory_id/{{
sim_inventory_id }}"
    method: PATCH
    headers:
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": null,
        "service_id": null,
        "item_state": "Available"
      }
    status_code: 200
  ignore_errors: True
  when: sim_inventory_id is defined

```

ActionPlans

```

- name: Remove ActionPlan from account
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.RemoveActionPlan",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Id": "ActionPlan_Monthly_{{ service_uuid }}_{{
product_id }}"
        }]
      }
    status_code: 200
  ignore_errors: True

```

□□□□□

```
- name: Release payment authorization
  uri:
    url: "http://localhost:5000/crm/payments/release/{{
authorization_id }}"
    method: POST
    headers:
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "metadata": {
          "release_reason": "provisioning_failed"
        }
      }
    return_content: yes
  ignore_errors: True
  when: authorization_id is defined
```

□□□□□□□□

□□□□□□□□

1. □□□□□□□□
2. □□□□□□ playbook□□□ action: "deprovision"
3. Playbook □□□□□□□□
4. □□□□□□□□
5. □□□□□□□□

□□ **API** □□□

```
POST /crm/provision/run_playbook
{
  "product_id": 123,
  "customer_id": 456,
  "service_id": 789,
  "action": "deprovision"
}
```

目標

ステップ 1 OCS のインストール

- OCS のインストール
- インストール
- インストール OCS のインストール
- インストール

ステップ 2 インストール

- OCS のインストール
- インストール
- インストール
- インストール OCS のインストール
- インストール

ステップ 3 インストール

- OCS のインストール
- インストール
- インストール
- インストール
 - インストール
 - インストール
 - インストール OCS のインストール

目標

1. インストール ignore_errors

インストール - インストール

2. インストール when インストール

インストール

when: service_creation_response is defined

3. 環境構築

環境構築の準備作業

```
- name: Print all vars for debugging
  debug:
    var: hostvars[inventory_hostname]
```

4. 環境構築

環境構築の準備作業

- 環境構築
- OCS のインストール
- 環境構築

5. 環境構築

環境構築の準備作業

6. 環境構築

OCS のインストール `ReloadScheduler: true` CGRateS のインストール

環境構築

環境構築 playbook の実行

- 環境構築
- 環境構築
- 環境構築
- 環境構築
- 環境構築

環境構築 API

- CRM のインストール
- OCS のインストール
- 環境構築
- 環境構築
- 環境構築

Ansible

Ansible `block/rescue` blocks allow you to group tasks into blocks and handle errors gracefully. This is useful for ensuring that your playbook can continue to run even if a task fails.

Example

```
- name: Main block
  block:
    - name: Step 1: Create account
      uri: ...
      register: response

    - name: Verify step 1
      assert:
        that:
          - response.status == 200

    - name: Step 2: Create service
      uri: ...

  rescue:
    # Handle errors gracefully
    # 1. Log the error
    # 2. Notify the user
    # 3. Playbook continues
    - name: Cleanup partial changes
      uri: ...
```

For more information, see the `concepts_provisioning` page.

OmniCRM API

OmniCRM 提供 RESTful API - 用于集成 UI 和移动应用

OmniCRM 提供 RESTful API 用于集成 UI 和移动应用

API 使用 RESTful API 格式，支持 JWT 令牌 API 调用 IP 限制

API 使用 Swagger 格式，提供 API 文档 API 调用

API 调用 URL 格式

<<https://yourcrm/crm/docs/>>

□□□□□□

OmniCRM □□□□□□□□□□□□□□□□□□□□

1. **JWT Bearer** □□ - □□□□□□□□□□Web UI□□□□□□

2. **API** -

3. **IP** -

JWT Bearer

JWT

- Web UI
-
-

JSON POST `/crm/auth/login`

```
{
  "email": "youruser@yourdomain.com",
  "password": "yourpassword"
}
```

API `token` JSON

`refresh_token`

Swagger `/auth/login` Try it out

Swagger 的“Authorize”按钮“Value”输入框
Bearer 输入框“Authorize”

API 32

API 32

API 32

API

- API 32
- API 32
- API 32
- API 32 cron

API 32

API 32 `crm_config.yaml` API 32 32
API 32 `X-API-KEY` API 32

API 32

API 32 `crm_config.yaml` API 32

```
api_keys:
  your-secure-api-key-here-minimum-32-chars:
    roles:
      - admin
    description: "Provisioning automation system"
  another-api-key-for-monitoring-system:
    roles:
      - view_customer
      - view_service
    description: "Monitoring and alerting"
```

API

X-API-KEY API

```
curl -X GET "https://yourcrm.com/crm/customers" \
-H "X-API-KEY: your-secure-api-key-here-minimum-32-chars"
```

Python

```
import requests

crm_url = 'https://yourcrm.com'
api_key = 'your-secure-api-key-here-minimum-32-chars'

headers = {
    "Content-Type": "application/json",
    "X-API-KEY": api_key
}

# 
response = requests.get(crm_url + '/crm/customers',
headers=headers)
for customer in response.json()['data']:
    print(customer)
```

- API (openssl rand -base64 48)

- 管理 API 端
- 在 `description` 中管理 API 端
- 管理 API 端
- 管理 API 端
- 管理 API 端

IP 管理

IP 管理 IP 地址管理 API 端

管理

- 管理
- 在 VLAN 中管理
- 在 Ansible 中管理

在 IP 中管理

管理 IP 地址 `crm_config.yaml`

```
ip_whitelist:
  - 192.168.1.100
  - 10.0.0.0/24
  - 172.16.50.10
```

管理

- 管理 IP 地址
- 管理 IP 地址
- 管理 IP 地址
- 管理 IP 地址
- 管理 IP 地址

Python API 端

管理 JWT 管理

```
import requests

crm_url = 'https://yourcrm.com'
session = requests.Session()

print("Provisioning data to server: " + str(crm_url))

headers = {
    "Content-Type": "application/json"
}

# 000000000000
response = session.post(crm_url + '/crm/auth/login', json={
    "email": "youruser@yourdomain.com",
    "password": "yourpassword"
}, headers=headers)

print(response.status_code)
print(response.json())
assert response.status_code == 200

headers['Authorization'] = 'Bearer ' + response.json()['token']
print("Authenticated to CRM successfully")

# 0000
response = session.get(crm_url + '/crm/customers',
headers=headers)
for customer in response.json()['data']:
    print(customer)
```

API ☐ ☐ ☐ ☐

Monitoring & Metrics

□□□□ Prometheus □□□ /crm/metrics □□□□□□□□□□

□□□□

- System Architecture - □□□□□□□□□□
- Monitoring & Metrics - □□□□□□□□□ API□□□□□□□□
- Authentication Flows - □□□□□□□□□□
- RBAC - □□□□□□□□□□

CRM 00 / 00000000

11

Ansible のインストールと設定 (Ubuntu 20.04 LTS)
 <guide_product_lifecycle>

```

SIM SIM <concepts_sim_provisioning>

```

--	--	--	--	--	--	--

--	--	--	--	--	--	--	--

“ ”

□ □

OmniCRM

○○○○○○○○○○○○○○○“○○○○”○“○○○○○○○○”○○ OmniCRM ○○○○○○○○○○○○○○○○○○○○○○○○○○○○○
○○○○○○○○○○○○○○○○

OmniCRM

[illegible]

環境構築

環境構築の準備作業を行います。

OmniCRM のインストールを行います。

インストールの準備作業を行います。

インストール時に必要な環境変数を設定します。IP アドレスは、`—`で指定します。

インストール

インストールの準備作業を行います。

OmniCRM のインストールを行います。インストール時に必要な環境変数を設定します。

インストールの準備作業を行います。

確認

インストールが完了したことを確認します。

確認作業を行います。

- インストールが完了したことを確認します。CRM のインストールが完了したことを確認します。
- インストールが完了したことを確認します。インストールが完了したことを確認します。
- インストールが完了したことを確認します。SIM のインストールが完了したことを確認します。

```
['SIM Card', 'Phone Number']  CRM のインストールが完了したことを確認します。
<administration_inventory>  CRM のインストールが完了したことを確認します。
```

- Ansible Playbook を実行します。Play <concepts_ansible> を実行します。Ansible のインストールが完了したことを確認します。service_id のインストールが完了したことを確認します。ICCID のインストールが完了したことを確認します。

MSISDN

category UI

- **standalone** - 標準パッケージ
- **addon** - 追加パッケージ
- **bundle** - 標準パッケージと追加パッケージのセット
- **promo** - プロモーション

標準パッケージと追加パッケージのセットを Ansible playbook で定義

例- `standalone` パッケージと追加パッケージ - `addon` と `bundle` パッケージと追加パッケージ - 標準パッケージ
 の playbook を定義 - 標準パッケージと追加パッケージのセットを定義

標準 UI パッケージ/追加パッケージ

例

`service_type` パッケージ

標準パッケージ

- **mobile** - 標準パッケージ
- **fixed** - 標準パッケージ
- **fixed-voice** - 標準パッケージ VoIP
- **hotspot** - 標準パッケージ
- **dongle** - USB 標準パッケージ
- **voice** - 標準パッケージ
- **data** - 標準パッケージ

標準パッケージと追加パッケージのセット

- 標準パッケージ
- 追加パッケージ
- 標準パッケージ
- 追加パッケージ

標準 `mobile` パッケージと追加パッケージ `fixed` パッケージ

3.   -         
4.   - Ansible playbook    
5.   -     

-   -      
-   -     
-  -    

-   -    FontAwesome 
-   -     
-  /   -   
-   -    \$  
-   -         

□□□□□

- 註冊
 - 註冊 - 註冊日期/月/日
 - 註冊 - 註冊地點
 - 註冊 - 註冊日期30 日內365 日內
- 註冊
 - 註冊 - 註冊日期
 - 註冊 - 註冊地點
- 註冊
 - 註冊 - 註冊日期 ID 註冊
 - 註冊 - 註冊日期

□□□□□□

Ansible

□□□□□

- **Play**
 - Ansible playbook `***.yaml` `***`
 - `*** OmniCRM-API/Provisioners/plays/ ***`
 - `*****`
- **JSON** `***`
 - `*** JSON *** Ansible playbook ***`
 - `*****`
 - Playbook `***** customer_id product_id service_id access_token`
- `*****`
 - `*****`
 - `***SIM *****IPv4 ***`
 - `*/*****`
 - `***** ID ***** playbook`

--	--	--	--	--	--	--

□ □

□□□□□

- □□□□□
 - □□□□□□□□□□/□□
 - □□□□□□□□
 - □□□□□□□□□□
- □□□□□
 - □□□□□□□□□□□□/□□
 - □□□□□□□□□□
 - □□□□□□□□□□□□□□
- □□□□□□□□□□
 - □□ - □□□□□□□□□□□□
 - □□□□ - □□□□□□□□□□
 - □□□□□□□□

□□□□□

- □□□□□
 - □ - □□□□
 - □□□□ - □□□□□□□“_clone”□□□□□□
 - □□□□ - □□□□□□□□
- □□/□□□□□□
 - □□ - □□□□□□□□
 - □□□□ - □□□□□□□□□□□□□□□□□□□

Table

Table with 10 columns and 1 row. The columns are: product_id, product_name, product_slug, category, service_type, comment, icon, retail_cost, wholesale_cost, tax_percentage.

Table

- **product_id** - Product ID
- **product_name** - Product Name (UI)
- **product_slug** - Product Slug (URL & API)
- **category** - Product Category (UI)
 - **standalone** - Standalone Product
 - **addon** - Addon Product
 - **bundle** - Bundle Product
 - **promo** - Promo Product
- **service_type** - Service Type (UI)
- **comment** - Comment
- **icon** - Icon (UI) (FontAwesome icons: fa-solid fa-sim-card)

Table

- **retail_cost** - Retail Cost (???)
- **wholesale_cost** - Wholesale Cost
- **retail_setup_cost** - Retail Setup Cost
- **wholesale_setup_cost** - Wholesale Setup Cost
- **tax_percentage** - Tax Percentage (10% or 12.5%)

概要

このサービスは、お客様の利用状況に応じて、自動的に料金を変更します。

- 利用量が 10%を超えると \$50.00 → \$5.00
- 利用量が 0%になると \$0.00
- サービスが終了すると \$0.00

利用条件

- **enabled** - サービスが有効かどうか。false は無効
- **residential** - 住宅用かどうか
- **business** - 商用かどうか
- **customer_can_purchase** - お客様が購入できるかどうか。true は購入可能、false は不可
- **available_from** - サービスが利用可能な開始日時
- **available_until** - サービスが利用可能な終了日時

契約

- **contract_days** - サービスの契約期間。30 日以上 365 日以内、0 日以内
- **auto_renew** - 自動更新かどうか
 - **prompt** - 自動更新の通知
 - **true** - 自動更新
 - **false** - 自動更新しない

□□□□□□

□□□□□□ **SIM**□

```
{
  "product_id": 1,
  "product_slug": "Mobile-SIM",
  "product_name": "Mobile SIM Only",
  "category": "standalone",
  "service_type": "mobile",
  "provisioning_play": "play_psim_only",
  "provisioning_json_vars": "{\"iccid\": \"\", \"msisdn\": \"\"}",
  "inventory_items_list": "['SIM Card', 'Mobile Number']",
  "retail_cost": 0,
  "retail_setup_cost": 0,
  "wholesale_cost": 3,
  "wholesale_setup_cost": 1,
  "contract_days": 0,
  "residential": true,
  "business": true,
  "enabled": true,
  "customer_can_purchase": true,
  "icon": "fa-solid fa-sim-card",
  "features_list": "['Australian Phone Number (04xxx)', 'Fastest speeds', 'Best coverage', 'Roaming on the Mainland']",
  "terms": "Must be activated within 6 months. All credit lost if service is not used for 12 months.",
  "comment": "Physical SIM card for use with Mobile Phones"
}
```

XXXXXXXXXXXXXXXXXXXXSIMXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXX

```
{
  "product_slug": "norfone-mobile-prepaid-mini",
  "product_name": "Norfone Mini Plan",
  "category": "addon",
  "service_type": "mobile",
  "provisioning_play": "play_topup_charge_then_action",
  "provisioning_json_vars": "",
  "inventory_items_list": "[]",
  "retail_cost": 30,
  "retail_setup_cost": 0,
  "wholesale_cost": 5.84,
  "contract_days": 30,
  "residential": true,
  "business": false,
  "enabled": true,
  "customer_can_purchase": true,
  "auto_renew": "prompt",
  "icon": "fa-solid fa-sim-card",
  "features_list": "['8GB of Ultra fast data', 'Unlimited Calls &
Texts to Norfone users', '100 Minutes of Talk to Australia', '100
SMS to Australia', '30 Day Expiry']",
  "terms": "Credit expires after 30 days. Once data, voice or sms
is used up, you will need to top up to continue using the
service.",
  "comment": "Our smallest plan for light users"
}
```

XX/XXXX

□□□□□□□□□□

```
{
  "product_slug": "Bundle-Seniors",
  "product_name": "Seniors Bundle",
  "category": "bundle",
  "service_type": "fixed",
  "provisioning_play": "play_seniors_package",
  "provisioning_json_vars": "{\"IPTV_Service_ID\":
\\\"SeniorBundle\\\"}\",
  "inventory_items_list": \"['Modem Router']\",
  "retail_cost": 30,
  "retail_setup_cost": 0,
  "wholesale_cost": 10,
  "wholesale_setup_cost": 11,
  "contract_days": 180,
  "residential": true,
  "business": false,
  "enabled": true,
  "icon": "fa-solid fa-person-walking-with-cane",
  "features_list": \"['20Mbps Download', '5Mbps Upload', 'Unlimited
Data', 'Home Voice', 'TV: Extra +£5 per month', '£60 Installation
Fee']\",
  "terms": "6 Month Contract, must show senior citizen's card to
qualify",
  "comment": "20Mbps/2Mbps GPON Service + IPTV + Phone"
}
```

□□□□□□□□□□ playbook □□□□□□□□□□ + IPTV + □□□□□□□□□□□□□□□□□□□□□□□□

4. 商品 - 商品ID `inventory_items_list` 商品

商品

商品商品商品商品商品商品商品商品商品商品商品商品商品商品商品

商品商品商品

```
"provisioning_json_vars": "{ \"monthly_cost\": 50, \"data_gb\": 100 }"
```

API 商品

```
{
  "product_id": 10,
  "customer_id": 456,
  "monthly_cost": 45,
  "custom_param": "value"
}
```

Ansible `extra_vars` 商品

```
{
  "monthly_cost": 45,      # 商品
  "data_gb": 100,          # 商品 provisioning_json_vars
  "product_id": 10,        # 商品
  "customer_id": 456,      # 商品
  "custom_param": "value", # 商品
  "access_token": "eyJ...", # 商品
}
```

商品

商品

- `access_token` - API 商品 CRM 商品 JWT 商品 IP/API 商品
`refresh_token` 商品

- `initiating_user` - 發起者 ID
- 產品 ID `product_id` 客戶 ID `customer_id` 服務 ID `service_id`

總結

Inventory items list: `inventory_items_list: ["SIM Card", "Mobile Number"]`

1. **UI/API** 端 - 發起者 ID
2. 產品 ID - 產品 ID
3. **Playbook** 端 ID - 該 playbook 的 CRM API 端 ID

Inventory items list: SIM Card ID: 789 - Mobile Number ID: 101

該 playbook 的 SIM Card: 789 - Mobile Number: 101

Playbook 的 ID 在 CRM API 的 ICCID, IMSI, MSISDN 中

Ansible 總結

Ansible 的 `extravars` 是 Ansible playbook 的 `playbook` 的 `extravars`

Ansible 的 `variable_name` 是 playbook 的 `product_id` 的 `customer_id` 的 `monthly_cost`

總結

Inventory items list

- `product_id` - 產品 ID
- `customer_id` - 客戶 ID
- `service_id` - 服務 ID
- `access_token` - API 的 access token
- `provisioning_json_vars` - 產品 ID
- API 的 ID

API Parameters

API "£5" parameters: `service_id: 123` playbook parameters

- `product_id: 45`
- `customer_id: 456`
- `service_id: 123`
- `access_token:`
- `provisioning_json_vars`

Playbook parameters

1. **CRM API** `service_id`
2. **UUID**
3. **OCS** `UUID`
4. **CRM**

API Parameters

Parameters

- `product_id` -
- `customer_id` -
- `ID` `SIM Card` `Mobile Number`
- `access_token` - API

Parameters

- `product_id` -
- `customer_id` -
- `service_id` - **ID**
- `access_token` - API

`service_id` - playbook parameters

API

API endpoint for `playbook` is `POST /api/v1/playbook`

- `product_id` - ID of product
- `customer_id` - ID of customer
- `service_id` - ID of service
- `modem_router_id` - ID of Modem Router
- `access_token` - API access token

API endpoint for `play_seniors_package` is `POST /api/v1/play_seniors_package`

UI

UI endpoint for `playbook` is `POST /ui/v1/playbook`

UI endpoint for `OCS` is `POST /ui/v1/ocs` and `CGRateS` is `POST /ui/v1/cgrates`

API Response

API response for `playbook` is `200 OK` with `application/json` content type

API Request

API request for `UI` is `POST /ui/v1/playbook`

1. `product_id` - ID of product
 - `residential = true` and `business = false` for residential
 - `business = true` for business
2. `category` - ID of category
 - `category = standalone` for standalone
 - `category = addon` for addon
3. `bundle` - ID of bundle

- `enabled = true` - 可購取
- 可購取 `available_from` 至 `available_until` 日期 - 可購取
- `customer_can_purchase = true` 可購取 - 可購取

API

API 可購取API 可購取

- 可購取 (/crm/product/) - 可購取
- 可購取 (/crm/product/paginated) - 可購取

可購取 `include_disabled=true` 可購取

UI 可購取

- 可購取 - 可購取
- 可購取 - 可購取
- 可購取 - 可購取
- 可購取 - 可購取

可購取

可購取 可購取

1. 可購取 - 可購取 `service_type` 可購取
 - 可購取 `service_type = "mobile"` 可購取 `service_type = "mobile"` 可購取
 - 可購取
2. 可購取 - 可購取 `relies_on_list`
 - 可購取/可購取
 - 可購取
3. 可購取 - 可購取 `residential` 至 `business` 可購取

可購取

可購取 `service_type = "mobile"` 可購取

- `business = true` `category != "addon"`
- `business = true` `category = "addon"` `service_type = "mobile"`
-

--	--	--	--

[illegible]

□ □ □ □ □ □

- **service_id** - 服务ID
- **customer_id** - 客户ID
- **product_id** - 产品ID
- **service_name** - 服务名称
- **service_type** - 服务类型 VoIP IPTV 其他
- **service_uuid** - OCS/CGRateS 服务UUID
- **icon** - 服务图标 FontAwesome 图标

□ □ □ □ □ □ □

- **service_status** - 00000000000000000000
- **service_provisioned_date** - 00000000000000000000
- **service_active_date** - 0000000000000000
- **service_deactivate_date** - 00000000000000000000
- **contract end date** - 0000000000000000

□ □ □ □ □

- **retail_cost** - 00000000000000000000
- **wholesale_cost** - 00000000000000
- **service_billed** - 00000000000000000000true
- **service_taxable** - 000000000000000000true
- **invoiced** - 0000000000000000000000
- **promo_code** - 00000000000000000000

Attributes

- **service_visible_to_customer** - 是否向客户可见 `true`
- **service_usage_visible_to_customer** - 是否向客户可见/是否向客户可见 `true`

Playbooks

- **provisioning_play** - 部署 Ansible playbook
- **provisioning_json_vars** - 部署 playbook 的 JSON 变量
- **deprovisioning_play** - 取消部署 Ansible playbook
- **deprovisioning_json_vars** - 取消部署 playbook 的 JSON 变量

Fields

- **bundled_parent** - 是否向客户可见 `service_id`
- **site_id** - 站点    是否向客户可见/是否向客户可见

Metadata

- **service_notes** - 服务备注
- **created** - 创建时间
- **last_modified** - 最后修改时间

API/UI

API/UI

PATCH /crm/service/{service_id}

- service_name
- service_type
- service_status
- service_notes
- retail_cost
- wholesale_cost
- service_billed
- service_taxable
- service_visible_to_customer
- service_usage_visible_to_customer
- service_active_date
- service_deactivate_date
- contract_end_date
- icon
- promo_code
- site_id

ステップ 1 準備

API 呼び出しの準備

- `provisioning_play` - Ansible playbook を `play_psim_only`
- `provisioning_json_vars` - 変数定義の JSON ファイル
- `customer_id` - 顧客 ID
- `product_id` - 製品 ID
- `service_id` - サービス ID
- `access_token` - API 呼び出し用の JWT トークン

ステップ 2 実行

API 呼び出しの実行

1. `provisioning_json_vars` ファイルを確認
2. 変数定義を確認
3. 実行コマンド
◦ `access_token` - API 呼び出し用の JWT トークン
◦ `initiating_user` - 実行ユーザー ID
◦ `customer_id` `product_id` `service_id`
4. 実行結果 - `{inventory_type: inventory_id}` を確認

実行結果

```
{
  "customer_id": 123,
  "product_id": 456,
  "service_id": 789,
  "SIM Card": 1001,
  "Mobile Number": 1002,
  "monthly_cost": 30,
  "data_gb": 50,
  "access_token": "eyJ...",
  "initiating_user": 5
}
```

顧客 ID
製品 ID
サービス ID
変数 provisioning_json_vars
変数 provisioning_json_vars
API 呼び出し用の JWT トークン
実行ユーザー ID

ステップ 3 確認

Provisioner Provision

- `provision_id` - Provisioner ID
- `provisioning_play` - playbook name
- `provisioning_json_vars` - Provisioner JSON vars
- `task_count` - playbook task count in YAML
- `provisioning_status` - Provisioner status 1 = Success 0 = Failure 2 = Pending 3 = Error
- `product_id` `customer_id` `service_id` - Provisioner metadata

4. playbook

API

1. `OmniCRM-API/Provisioners/plays/{playbook_name}.yaml` playbook YAML
2. `ansible_runner.run()`
 - `playbook` - playbook YAML
 - `extravars` - Ansible extras vars
 - `inventory` - `'localhost,'`
 - `event_handler` - Provisioner event handler
3. playbook

5. `ProvisioningEventHandler`

Ansible `Provision_Event`

- `event_name` - playbook name
- `event_number` - event number
- `provisioning_status` - Provisioner status
 - 0 = Success
 - 1 = Failure
 - 2 = Pending
 - 3 = Error (if `ignore_errors: true` in playbook)
- `provisioning_result_json` - Provisioner result JSON

6 Ansible playbook Provisioners/plays/*.yaml

Ansible playbook

1. `access_token` - `access_token` `/crm/product/product_id/{{ product_id }}` GET
2. `customer_id` - `/crm/customer/customer_id/{{ customer_id }}` GET
3. `inventory_id` - `/crm/inventory/inventory_id/{{ inventory_id }}` GET
ICCID MSISDN
4. API
 - HSS
 - IMS IP
 - CGRateS/OCS
 - ENUM
 -
5. `transaction` - `/crm/transaction/` POST
6. `retail_setup_cost` - `OCS/CGRateS` POST
7. **OCS** - `OCS/CGRateS` POST UUID
8. `Actions` `ActionPlans` - `OCS/CGRateS`
9. `service` - `/crm/service/` PUT/POST CRM

```
{
  "customer_id": 123,
  "product_id": 456,
  "service_name": "Mobile SIM - 0412345678",
  "service_uuid": "generated-uuid-for-ocs",
  "service_status": "Active",
  "service_type": "mobile",
  "retail_cost": 30,
  "wholesale_cost": 5,
  "provisioning_play": "play_psim_only",
  "provisioning_json_vars": "{...}"
}
```

10. PATCH - `/crm/inventory/inventory_id/{{ inventory_id }}` PATCH
 PATCH

11. -

7

playbook

- `Provision.provisioning_status` 0
- `Provision.provisioning_status` 2
`crm_config.provisioning.failure_list`
- `ignore_errors: true` 3

CRM

`category: standalone`

- `customer_id` `product_id`
- SIM
- API PUT `/crm/service/`
- HSS OCS
- SIM

category: addon

- `customer_id` `product_id` `service_id`
-
- OCS
- OCS
-
-

category: bundle

-
-
- + +
-
- CRM
- + IPTV + VoIP

playbook

playbook

1. `OmniCRM-API/Provisioners/plays/{playbook_name}.yaml`
2. **Ansible** `extravars` `{{ variable_name }}`
3. **Authorization: Bearer {{ access_token }}** **API**
4. `rescue` `ignore_errors`
- 5.
- 6.
7. `fail`

Playbooks

playbook

- `customer_id` -
- `product_id` -

- `service_id` - 服務 ID/服務名稱
- `access_token` - CRM API 的 JWT 值
- `initiating_user` - 發起人
- 產品 ID `{{ inventory_type }}: inventory_id`
- 配置 `provisioning_json_vars` 參數
- 其他參數

Playbooks 參照  說明

- 查詢產品 `GET /crm/product/product_id/{{ product_id }}`
- 查詢客戶 `GET /crm/customer/customer_id/{{ customer_id }}`
- 查詢庫存 `GET /crm/inventory/inventory_id/{{ SIM_Card }}`
- 提交交易 `POST /crm/transaction/`
- 提交服務 `PUT /crm/service/`
- 更新服務 `PATCH /crm/service/{{ service_id }}`
- 更新庫存 `PATCH /crm/inventory/inventory_id/{{ inventory_id }}`

如何執行 **playbook** 呢

執行步驟如下

1. Playbook 參數 `customer_id` `product_id` `service_id` `access_token`
2. 查詢服務 `GET /crm/service/{{ service_id }}` 獲取 `service_uuid`
3. 查詢產品 `GET /crm/product/product_id/{{ product_id }}` 獲取產品名稱
4. 向 OCS 提交交易 `POST to CGRateS` 參數 `retail_cost`
5. 向 OCS 提交服務 `POST to CGRateS` 參數 `service_uuid`
6. 向 CRM 提交交易 `POST /crm/transaction/` 參數 `product_id`
7. 完成

執行 `Provision` 的 `Provision_Event` 參照  說明

OCS 參照

OCS 參照 `CGRateS` 參數 `retail_cost` 參照 CRM 參照 OCS 參照

- **பெயர்** - **பெயர்**DID **பெயர்**
- **பெயர்** - **பெயர்** MB **பெயர்**
- **பெயர்** - **பெயர்** GB **பெயர்**
- **பெயர்** - **பெயர்** \$30 **பெயர்** 10GB **பெயர்**
- **பெயர்** - **பெயர்**

CRM **பெயர்** OCS **பெயர்**

பெயர்

பெயர் OCS/CGRateS **பெயர்**

பெயர்

பெயர் UI **பெயர்** API **பெயர்**

1. **API** **பெயர்** - **பெயர்** GET /crm/service/{service_id} **பெயர்** UI **பெயர்**
2. **பெயர்** - API **பெயர்** service_uuid
3. **CGRateS API** **பெயர்** - **பெயர்** CGRateS **பெயர்**
 - i. **Get_Balance(service_uuid)** - **பெயர்** BalanceMap
 - **பெயர்** **பெயர்**
 - **பெயர்**ID **பெயர்** ID
 - **பெயர்** custom_Name_hr custom_Expiration custom_Description_String
 - ii. **Get_ActionPlans(service_uuid)** - **பெயர்**
4. **பெயர்** - CGRateS **பெயர்**

```

{
  "service_id": 123,
  "service_name": "Mobile Service",
  "service_uuid": "abc-123-def",
  "cgrates": {
    "BalanceMap": {
      "DATA": [{
        "ID": "DATA_10GB",
        "Value": 5368709120,
        "ExpirationDate": "2025-02-01T00:00:00Z",
        "custom_Name_hr": "10GB Data Pack",
        "custom_Expiration": "Feb 1, 2025",
        "custom_Description_String": "5 GB remaining"
      }],
      "VOICE": [{
        "ID": "VOICE_UNLIMITED",
        "Value": 999999999,
        "custom_Name_hr": "Unlimited Calls",
        "custom_Description_String": "Unlimited minutes"
      }],
      "MONETARY": [{
        "ID": "PREPAID_CREDIT",
        "Value": 25.50,
        "custom_Description_String": "$25.50 credit"
      }]
    },
    "ActionPlans": [...]
  }
}

```

5. UI -

- **ServiceUsage.js** - 3
- **UsageCard.js** -
- **UsageProgress.js** - /
-

BalanceMap

CGRateS

- **ID** - 唯一标识符 "DATA_10GB_2025_01"
- **Value** - 数值
 - 5368709120 = 5 GB
 - 3600 = 1 小时
 - 100 = 100 分钟
 - 25.50 = \$25.50
- **ExpirationDate** - 到期日期 ISO 8601 格式
- **Weight** - 权重
- **DestinationIDs** - 目标 ID ["AU", "INTERNATIONAL"]

CRM

- **custom_Name_hr** - 客户 ID
- **custom_Expiration** - 到期日期 "2025 年 1 月 15 日" "11 月"
- **custom_Description_String** - 描述字符串
 - "5 GB" "10 GB"
 - "60 分钟" "小时"
 - "50 分钟"
 - "\$25.50"

- **service_visible_to_customer** - 是否 visible 到客户
- **service_usage_visible_to_customer** - 是否 visible 到客户/是否 visible 到客户

- 日期/时间
- 字符串/数字
- 布尔值

API 說明

Web UI 說明

- 目前 3 個 ServiceUsage 說明
- 目前 GET /crm/service/{service_id} 的 CGRateS 說明
- 目前 1 個 ServiceUsage 說明

目前 1 個 ServiceUsage 說明

API 說明 / 說明

目前 1 個 ServiceUsage 說明 DID 說明 OCS 說明 Actions 說明
Action_ServiceUUID_ServiceName_WhatitDoes

目前 \$60 的 GPON 說明 1TB 說明 Action 說明

Action_kj49-adsf-1234-9742_60_GPON_1TB_MonthlyExpiry

1. 目前 \$0
2. 目前 CRM 的 HTTP POST 說明
3. 目前 1TB 說明 1 個

目前 MRC 說明 ActionPlan_{service_uuid}_Monthly_Charge 的 ActionPlan 說明 ActionPlan 說明

目前 12 個 MRC 說明

目前 Actions 的 ActionPlans 說明

目前 1 個 ServiceUsage 說明

API 說明

目前/目前 1 個 Action 說明

[[[ActionPlans]]]] Ansible [[[Action]]] ExecuteAction
[[[

[[[

[[[\$10 PAYG]]]]

[[[\$0

[[[/]]]]

ThresholdS [[[

[[[PAYG /]]]] \$0

[[[CRM]]] OCS [[[

[[[CRM API]]] OCS [[[Balances]]] ActionPlans [[[Actions]]]
ThresholdS

[[[CRM API]]] ActionPlans [[[Ansible Playbooks]]] CRM [[[
[[[ActionPlans]]]/]]] Ansible Playbooks [[[

[[[OCS]]] ActionPlans [[[

[[[ThresholdS]]]

[[[CRM]]] ActionPlans

ActionPlans [[[CRM]]]

[[[ActionPlans

[[[CRM]]] ActionPlans [[[

1. API 関数 - 関数 GET /crm/service/{service_id} API

- 関数呼び出し
- 関数 service_uuid OCS 関数
- 関数 CGRateS API 関数 UUID 関数 ActionPlans
- 関数 Get_ActionPlans(service_uuid) 関数 CGRateS

2. ActionPlan 関数 - 関数 ActionPlan 関数

```
{
  "ActionPlanId": "ServiceID_abc-123-
def__ProductID_456__...",
  "PlanName": "Monthly_Renewal_Plan",
  "NextExecTime": "2025-02-01T00:00:00Z",
  "custom_NextExecTime_hr": "in 11 days",
  "ActionPlanId_split_dict": {
    "ServiceID": "abc-123-def",
    "ProductID": 456,
    "CustomerID": 789,
    ...
  }
}
```

- ActionPlanId - 関数/関数/関数
- PlanName - 関数 playbook 関数
- NextExecTime - 関数 ActionPlan 関数 ISO 関数
- custom_NextExecTime_hr - 関数“11 関数”関数“2025 関数 2 関数 1 関数”
- ActionPlanId_split_dict - 関数 ActionPlanId 関数

3. UI 関数 - ActionPlans 関数 ActionPlansTable 関数

関数

- 関数 - 関数 ActionPlanId 関数 ProductID 関数
- 関数 - 関数 retail_cost
- 関数 - 関数 custom_NextExecTime_hr 関数
- 関数 - 関数

- 刪除 - 刪除所有/刪除指定???
- 刪除 - 刪除

刪除 **ActionPlans**

- 刪除“刪除”
- 刪除“刪除”

刪除 ActionPlans

刪除 UI 刪除 ActionPlans

刪除 **ActionPlan**

1. 刪除 ActionPlansTable 刪除“刪除”
2. 刪除“刪除”
3. 刪除 `DELETE /crm/oam/remove_action_plan/{action_plan_id}`
4. API 刪除 `ocs.Remove_ActionPlan()` 刪除 CGRateS 刪除 ActionPlan
5. 刪除“刪除 {service_id} 刪除 ActionPlan {ActionPlanId}”
6. ActionPlan 刪除

刪除

1. 刪除 ActionPlansTable 刪除“刪除”
2. 刪除“刪除”
3. 刪除
 - 刪除 ActionPlanId 刪除 product_id
 - 刪除
 - 刪除 playbook
 - 刪除
4. 刪除
5. 刪除

刪除

刪除 `auto_renew` 刪除

- `auto_renew = "true"` 刪除 - 刪除 ActionPlans

- `auto_renew = "prompt"` 自動 - 自動的に更新されます
- `auto_renew = "false"` 手動 - 手動で ActionPlans を更新します

この playbook は CGRateS から ActionPlan を取得します

- 各 ActionPlan の ID を `ActionPlanId`
- 各 ActionPlan の名前
- 各 ActionPlan の説明
- 各 ActionPlan の有効期限

ActionPlan の取得

ActionPlans を取得するための SQL 文

SQL

```
ServiceID_{service_uuid}__ProductID_{product_id}__CustomerID_{customer_id}
```

SQL

```
ServiceID_abc-123-def__ProductID_456__CustomerID_789__MonthlyRenewal
```

この CRM は

- 各 ActionPlan の名前
- 各 ActionPlan の説明
- 各 ActionPlan の有効期限
- 各 ActionPlan の更新日時

CRM から取得する `ActionPlanId_split_dict` の結果

各 ActionPlans の取得

この CRM から取得する ActionPlans の結果

ServiceView.js

- ActionPlans
-
-

-
-
-

- ActionPlans“”
-

ActionPlans

CRM

GET /crm/service/expiring?threshold=7

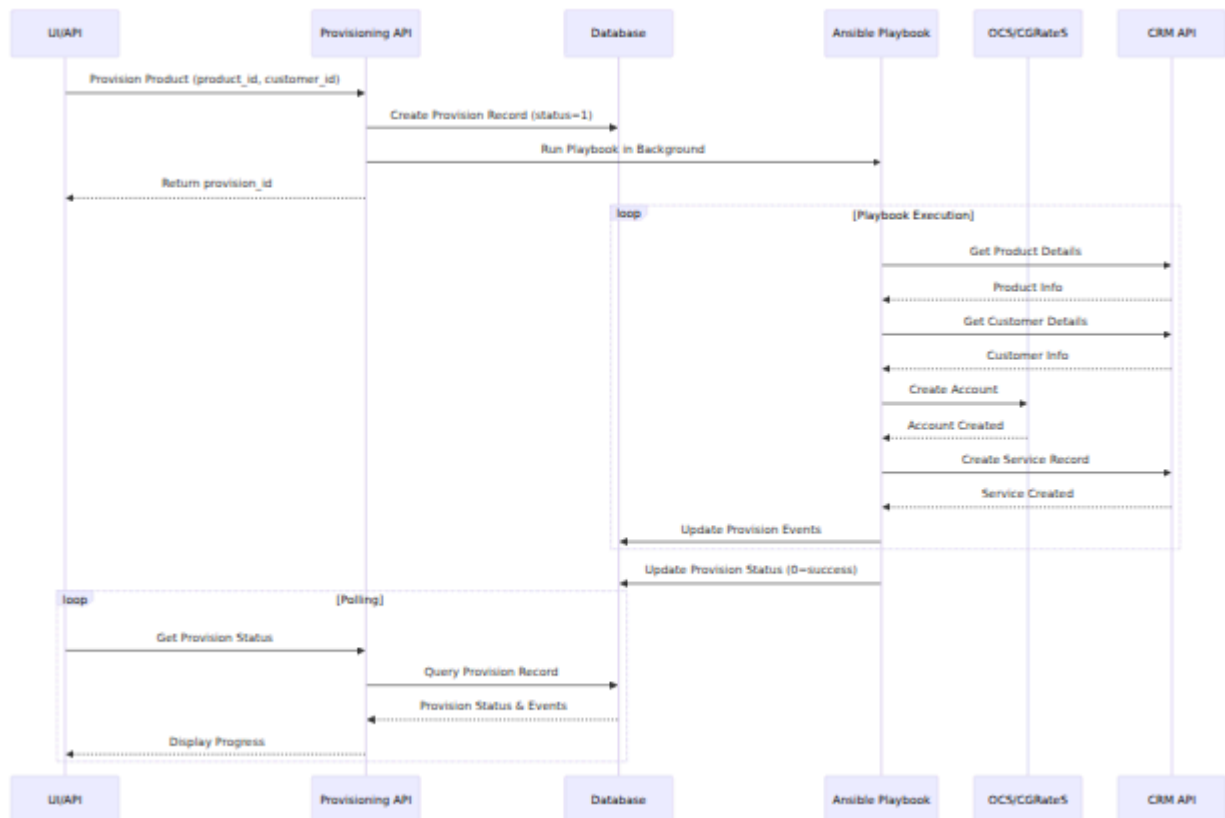
-
- ActionPlans
- ActionPlans

-
-
-
-



OmniCRM 的 **Ansible** 部署和运行流程

通过 `SIM` 概念 `<concepts_sim_provisioning>` 部署 SIM 卡，支持 eSIM。



11

```

##### Ansible #####
<guide_product_lifecycle>

```

11

OmniCRM 部署 部署 部署 Ansible 部署 部署
OCS/CGRateS部署API 部署

□ □ □ □ □ □ □ □ □ □ □ □ □ □

1. 開發 - 開發 UI/API 開發
2. 測試 - 測試 OCS 開發

--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- 000000 - 00000000
- 100000 - 00000000000000
- 20000 - 000 - 0000000000000000
- 3000 - 0000 - 0000000000 `ignore_errors: true`00000000

OmniCRM

--	--	--	--	--	--	--	--

[illegible]

□ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □

- provisioning_play - Ansible playbooks .yaml

- `provisioning_json_vars` - Ansible JSON vars
- `inventory_items_list` - list of items ['SIM Card', 'Mobile Number']
- `product_id` & `product_name` - product details

Example

```
{
  "product_id": 1,
  "product_slug": "Mobile-SIM",
  "product_name": "Mobile SIM Only",
  "provisioning_play": "play_psim_only",
  "provisioning_json_vars": "{\"iccid\": \"\", \"msisdn\": \"\"}",
  "inventory_items_list": ["SIM Card", "Mobile Number"],
  "retail_cost": 0,
  "retail_setup_cost": 0,
  "wholesale_cost": 3,
  "wholesale_setup_cost": 1
}
```

Implementation

Steps

1. Run `provisioning_play` playbook

Path: `OmniCRM-API/Provisioners/plays/play_psim_only.yaml`

2. Pass `extra_vars` as

- i. `provisioning_json_vars` as `{"iccid": "", "msisdn": ""}`
- ii. `OmniCRM-API` details
- iii. `product_id` & `customer_id`
- iv. `access_token` & `refresh_token`

3. Run `inventory_items_list` playbook

Path: `OmniCRM-API`

- **SIM** - SIM Card
- **Mobile Number** - Mobile Number

Inventory ID is stored in `extra_vars` dictionary

```
extra_vars = {
    "product_id": 1,
    "customer_id": 456,
    "SIM Card": 789,          # SIM Card inventory_id
    "Mobile Number": 101,    # Mobile Number inventory_id
    "iccid": "",             # iccid provisioning_json_vars
    "msisdn": "",            # msisdn provisioning_json_vars
    "access_token": "eyJ..."
}
```

4. `hostvars[inventory_hostname]` dictionary in **Ansible**

Inventory dictionary

```
- name: Get inventory_id for SIM Card
  set_fact:
    inventory_id_sim_card: "{{ hostvars[inventory_hostname]
['SIM Card'] | int }}"
  when: "'SIM Card' in hostvars[inventory_hostname]"
```

Inventory dictionary

Inventory ID is stored in API dictionary

```
- name: Get SIM Card Details from Inventory
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ inventory_id_sim_card }}"
  method: GET
  headers:
    Authorization: "Bearer {{ access_token }}"
  return_content: yes
  register: sim_card_response

- name: Extract ICCID and IMSI from inventory
  set_fact:
    iccid: "{{ sim_card_response.json.iccid }}"
    imsi: "{{ sim_card_response.json.imsi }}"

- name: Get Phone Number Details from Inventory
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ inventory_id_phone_number }}"
  method: GET
  headers:
    Authorization: "Bearer {{ access_token }}"
  return_content: yes
  register: phone_number_response

- name: Extract MSISDN
  set_fact:
    msisdn: "{{ phone_number_response.json.msisdn }}"
```

□ □ □ □ □ □ □ □ □ □ □ □ □

- HSS SIM IMSI
-
-
-

□□□□□□□□ **SIM** □□

```
play_psim_only.yaml
```

- name: Get Product information from CRM API
 - uri:
 - url: "{{ crm_config.crm.base_url }}/crm/product/product_id/{{ product_id }}"
 - method: GET
 - headers:
 - Authorization: "Bearer {{ access_token }}"
 - return_content: yes
 - register: api_response_product
- name: Set package facts from product
 - set_fact:
 - package_name: "{{ api_response_product.json.product_name }}"
 - package_comment: "{{ api_response_product.json.comment }}"
 - setup_cost: "{{ api_response_product.json.retail_setup_cost }}"
 - monthly_cost: "{{ api_response_product.json.retail_cost }}"
- name: Set inventory_id_sim_card if SIM Card was selected
 - set_fact:
 - inventory_id_sim_card: "{{ hostvars[inventory_hostname]['SIM Card'] | int }}"
 - when: "'SIM Card' in hostvars[inventory_hostname]"
- name: Set inventory_id_phone_number if Mobile Number was selected
 - set_fact:
 - inventory_id_phone_number: "{{ hostvars[inventory_hostname]['Mobile Number'] | int }}"
 - when: "'Mobile Number' in hostvars[inventory_hostname]"
- name: Get SIM Card details from inventory
 - uri:
 - url: "{{ crm_config.crm.base_url }}/crm/inventory/inventory_id/{{ inventory_id_sim_card }}"
 - method: GET
 - headers:
 - Authorization: "Bearer {{ access_token }}"
 - return_content: yes
 - register: sim_inventory_response
- name: Get Phone Number details from inventory
 - uri:


```
url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ inventory_id_phone_number }}"
method: GET
headers:
  Authorization: "Bearer {{ access_token }}"
return_content: yes
register: phone_inventory_response
```

- name: Extract values from inventory

```
set_fact:
  iccid: "{{ sim_inventory_response.json.iccid }}"
  imsi: "{{ sim_inventory_response.json.imsi }}"
  msisdn: "{{ phone_inventory_response.json.msisdn }}"
  ki: "{{ sim_inventory_response.json.ki }}"
  opc: "{{ sim_inventory_response.json.opc }}"
```

- name: Provision subscriber on HSS

```
uri:
  url: "http://{{ hss_server }}/subscriber/{{ imsi }}"
  method: PUT
  body_format: json
  body:
    {
      "imsi": "{{ imsi }}",
      "msisdn": "{{ msisdn }}",
      "ki": "{{ ki }}",
      "opc": "{{ opc }}",
      "enabled": true
    }
  status_code: 200
```

- name: Assign inventory to customer

```
uri:
  url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ inventory_id_sim_card }}"
  method: PATCH
  headers:
    Authorization: "Bearer {{ access_token }}"
  body_format: json
  body:
    {
      "customer_id": {{ customer_id }},
      "item_state": "Assigned"
```

```
}  
status_code: 200
```

□□□□□□□□□□

1. □□□□□□ `provisioning_play: "play_psim_only"`
2. □□□□ `inventory_items_list: ['SIM Card', 'Mobile Number']`
3. □□□□□□□□□□□□□□□□
4. □□ ID □□ `extra_vars` □□□□□□
5. □□□ API □□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□
7. □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ Ansible □ `block` □ `rescue` □□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

□□ `play_psim_only.yaml`□

```

- name: OmniCore Service Provisioning 2024
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Main block
      block:
        # --- 0000 ---
        - name: Get Product information
          uri: ...

        - name: Create account in OCS
          uri: ...

        - name: Provision subscriber on HSS
          uri: ...

        - name: Create service record
          uri: ...

        # ... 00000000 ...

      rescue:
        # --- 0000 ---
        # 000000000000
        # 1. 000000000000
        # 2. action == "deprovision"000000

        - name: Get Inventory items linked to this service
          uri:
            url: "{{ crm_config.crm.base_url
            }}/crm/inventory/customer_id/{{ customer_id }}"
            method: GET
            register: inventory_api_response
            ignore_errors: True

        - name: Return inventory to pool
          uri:
            url: "{{ crm_config.crm.base_url
            }}/crm/inventory/inventory_id/{{ item.inventory_id }}"
            method: PATCH
            body_format: json

```

```

    body:
      service_id: null
      customer_id: null
      item_state: "Used"
with_items: "{{ inventory_api_response.json.data }}"
ignore_errors: True

- name: Delete Account from Charging
  uri:
    url: "http://{{ crm_config.ocs.OCS }}/jsonrpc"
    method: POST
    body:
      {
        "method": "ApierV1.RemoveAccount",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}"
        }]
      }
    ignore_errors: True

- name: Delete Attribute Profile
  uri:
    url: "http://{{ crm_config.ocs.OCS }}/jsonrpc"
    method: POST
    body:
      {
        "method": "APIerSv1.RemoveAttributeProfile",
        "params": [{
          "ID": "ATTR_ACCOUNT_{{ service_uuid }}"
        }]
      }
    ignore_errors: True

- name: Remove Resource Profile
  uri: ...
  ignore_errors: True

- name: Remove Filters
  uri: ...
  ignore_errors: True

- name: Deprovision Subscriber from HSS
  uri:

```

```

      url: "{{ item.key }}/subscriber/{{
item.value.subscriber_id }}"
      method: DELETE
      loop: "{{ hss_subscriber_data | dict2items }}"
      ignore_errors: True
      when:
        - deprovision_subscriber | bool == true

- name: Patch Subscriber to Dormant State
  uri:
    url: "{{ item.key }}/subscriber/{{
item.value.subscriber_id }}"
    method: PATCH
    body:
      {
        "enabled": true,
        "msisdn": "9999{{ imsi[-10:] }}", # []
        "ue_ambr_dl": 9999999, # []
        "ue_ambr_ul": 9999999
      }
    loop: "{{ hss_subscriber_data | dict2items }}"
    when:
      - deprovision_subscriber | default(false) | bool ==
false

# []
- name: Set status to "Success" if Manual deprovision /
Fail if failed provision
  assert:
    that:
      - action == "deprovision"

```

[][][][][][][][]

1. []

[][][][][][][][]

- [] [] block [] rescue []
- [] action: "deprovision" [] rescue

2. []

rescue

- OCS
-
-
- HSS
-

“”

-
-
-
-

3. ignore_errors

ignore_errors: True

-
-
-

“ OCS ”

- OCS
-
-
- HSS

4.

rescue

```
- name: Set status to "Success" if Manual deprovision / Fail if
failed provision
  assert:
    that:
      - action == "deprovision"
```

#####

- `action == "deprovision"` ##### 0
- `action` ##### `!= "deprovision"` ##### 2

#####

5.

#####

```
- name: Deprovision Subscriber from HSS
  uri: ...
  when:
    - deprovision_subscriber | bool == true

- name: Patch Subscriber to Dormant State
  uri: ...
  when:
    - deprovision_subscriber | default(false) | bool == false
```

#####

- ##### SIM ##### `deprovision_subscriber: true`
- ##### SIM ##### HSS ##### `deprovision_subscriber: false`

#####

#####

```
{
  "product_id": 1,
  "customer_id": 456,
  "provisioning_play": "play_psim_only"
}
```

rescue

```
{
  "service_id": 123,
  "service_uuid": "Service_abc123",
  "action": "deprovision",
  "provisioning_play": "play_psim_only"
}
```

rescue

action: "deprovision"

provisioning_json_vars

```
{
  "provisioning_json_vars": "{\"monthly_cost\": 50,
  \"data_limit_gb\": 100}"
}
```



```
{
  "product_id": 1,
  "customer_id": 456,
  "monthly_cost": 45,
  "data_limit_gb": 150
}
```

extra_vars

-
-
-

```
{
  "product_id": 10,
  "product_name": "Extra 10GB Data",
  "provisioning_play": "play_local_data_addon",
  "provisioning_json_vars": "{\"data_gb\": 10}",
  "inventory_items_list": "[]"
}
```

```
extra_vars = {
  "product_id": 10,
  "customer_id": 456,
  "service_id": 123, #
  "data_gb": 10,
  "access_token": "eyJ..."
}
```

API 実装

API 実装の概要

- UI 実装
- データベース実装
- API 実装 `PUT /crm/provision/`

API 実装

API 実装の概要

1. UI 実装

API 実装の概要

- `provisioning_play` - Ansible プレイ
- `inventory_items_list` - データベースから `['SIM Card', 'Mobile Number']` を取得
- `provisioning_json_vars` - JSON 変数

2. データベース実装

`inventory_items_list` はデータベースからデータを取得する関数。データベースからデータを取得する関数。

3. API 実装

JavaScript で `PUT /crm/provision/` を実装

```
PUT /crm/provision/
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "product_id": 42,
  "customer_id": 123,
  "SIM Card": 5001,
  "Mobile Number": 5002
}
```

4. API 测试

测试

- 测试Bearer 令牌API 接口 IP 地址
- 测试CREATE_PROVISION 接口
- 测试initiating_user
- 测试
- 测试OmniCRM-API/Provisioners/plays/play_psim_only.yaml

5. 测试

测试测试测试测试测试

```
# 测试
product_vars = json.loads(product['provisioning_json_vars'])
# 测试
request_vars = request_body_json
# 测试
system_vars = {
    'product_id': 42,
    'customer_id': 123,
    'access_token': auth_token, # 测试
    'initiating_user': 7
}
# 测试
extra_vars = {**product_vars, **request_vars, **system_vars}
```

6. 测试

测试测试 1测试测试

```
provision = {
    'provision_id': 456,
    'customer_id': 123,
    'product_id': 42,
    'provisioning_play': 'play_psim_only',
    'provisioning_json_vars': json.dumps(extra_vars),
    'provisioning_status': 1, # 成功
    'task_count': 85,
    'initiating_user': 7,
    'created': '2025-01-10T14:30:00Z'
}
```

7. 測試

測試以下配置

- 測試 `play_psim_only.yaml`
- 測試 `extra_vars`
- 測試 ID 456 的 provisioning
- 測試 provisioning 的任務數量

8. API 測試

測試 UI 的 `provision_id`

```
{
  "provision_id": 456,
  "provisioning_status": 1,
  "message": "Provisioning job created"
}
```

9. UI 測試

UI 測試 3 個 GET `/crm/provision/provision_id/456` 的測試

```

{
  "provision_id": 456,
  "provisioning_status": 1,
  "task_count": 12,
  "provisioning_result_json": [
    {
      "event_number": 1,
      "event_name": "Get Product information from CRM API",
      "provisioning_status": 0,
      "timestamp": "2024-01-15T10:30:05"
    },
    {
      "event_number": 2,
      "event_name": "Assign SIM Card from inventory",
      "provisioning_status": 1,
      "timestamp": "2024-01-15T10:30:07"
    }
  ]
}

```

10. Ansible 環境構築

環境構築の目的

- 環境構築の自動化を目的として `Provision_Event` を作成
- 環境構築の自動化を目的として 0=成功 2=失敗 3=エラーを返す JSON を作成
- UI 環境構築の自動化を目的として 環境構築の自動化を目的として
- 環境構築の自動化を目的として 環境構築の自動化を目的として

UI 環境構築

環境構築の自動化を目的として 1=成功を返す

- 環境構築の自動化を目的として 環境構築の自動化を目的として
- 環境構築の自動化を目的として 環境構築の自動化を目的として
- 環境構築の自動化を目的として 環境構築の自動化を目的として

環境構築

環境構築の自動化を目的として 12 環境構築の自動化を目的として 8 環境構築の自動化を目的として

✓ CRM API 成功 ✓ 成功 ✓ 成功 SIM ICCID:
8991101200003204510 ✓ 成功 555-0123 & OCS/CGRateS 成功
... 成功 ...

11. 成功

成功

- provisioning_status: 0 - 成功
- provisioning_status: 2 - 成功

UI 成功

- 成功
- 成功 X 成功
- 成功 provisioning.failure_list

成功

成功

成功

- initiating_user 成功 JWT 成功 ID
- API 成功 ID
- IP 成功 ID

成功

成功

- CREATE_PROVISION 成功
- VIEW_OWN_PROVISION 成功

Ansible 成功 CRM API 成功

Ansible 成功 API 成功 CRM 成功
Bearer 成功

access_token 是 API 的访问令牌

1 Bearer

Web UI

1. `POST /crm/auth/login`
2. JWT `access_token` 15-30 `refresh_token`
3. Bearer
4. API `Authorization: Bearer ...`
- 5.
6. `access token` Ansible

□□□□□

```
Authorization Bearer [REDACTED]
API [REDACTED]
```

2 API X-API-KEY

□□□□ API □□□□□□□□

1. `PUT /crm/provision/` `X-API-KEY: your-api-key...`
2. API `crm_config.yaml`
3. **JWT**
- 4.
5. Ansible

□ □ □ □ □ □ □ □

API □□□□□□□□ JWT□□□□□ API □□□□□ JWT □□□□□□□□

- API 端
- 数据库端 admin 数据库端 JWT
- 数据库端 ID 端 JWT 端
- 数据库端 API 端

□□□□□

API 3 IP Whitelist JWT Ansible

3 IP Whitelist

API 3 IP Whitelist

1. IP Whitelist 192.168.1.100
2. API IP Whitelist `crm_config.yaml` `ip_whitelist`
3. JWT Token
4. Ansible
5. Ansible

API 3 IP Whitelist

API 3 IP Whitelist JWT Token

API 3 IP Whitelist

API 3 IP Whitelist

```
- name: Get Product Details
  uri:
    url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
    headers:
      Authorization: "Bearer {{ access_token }}"

- name: Create Service Record
  uri:
    url: "http://localhost:5000/crm/service/"
    method: PUT
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      customer_id: "{{ customer_id }}"
      service_name: "Mobile Service"
```

API 3 IP Whitelist

access_token 5-10 refresh_token 15-30

```
refresh_token = request.cookies.get('refresh_token')
run_playbook(playbook_path, extra_vars, provision_id,
refresh_token=refresh_token)
```

access_token

1. 401
2. refresh_token POST /crm/auth/refresh
3. access_token
- 4.

API IP 1-2

- 1.

-
- Ansible
- Provision
-
- API

- 2.

Ansible CRM API

- Bearer JWT refresh_token
- API IP access_token

- 3.

部署前需要部署的 API 部署前需要部署的

部署前

- 部署前/部署前 `Provision_Event` 部署
- 部署前部署前部署前部署前
- 部署前部署前
- UI 部署 `GET /crm/provision/provision_id/<id>` 部署

4. 部署

Ansible 部署前部署前

- 部署 API 部署前
- 部署 API 部署前
- 部署前SIM 部署IP 部署前部署前
- 部署 OCS/OCS 部署前
- 部署前
- 部署 CRM API 部署前部署前
- 部署前部署前
- 部署前部署前/SMS

5. 部署

Ansible 部署 `block` 部署 `rescue` 部署前部署前

- 部署前部署前 `rescue` 部署前部署前
- 部署前 3 部署 `ignore_errors: true` 部署前部署前
- 部署前YAML 部署前部署前部署前部署前部署前部署前

部署前部署前部署前

部署前 `play_simple_service.yaml` 部署前

```

- name: Simple Provisioning Play
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Main block
      block:
        - name: Get Product information from CRM API
          uri:
            url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
            method: GET
            headers:
              Authorization: "Bearer {{ access_token }}"
            return_content: yes
            validate_certs: no
            register: api_response_product

        - name: Set package facts
          set_fact:
            package_name: "{{ api_response_product.json.product_name
}}"

            setup_cost: "{{
api_response_product.json.retail_setup_cost }}"
            monthly_cost: "{{ api_response_product.json.retail_cost
}}"

        - name: Generate Service UUID
          set_fact:
            service_uuid: "Service_{{ 99999999 | random | to_uuid
}}"

        - name: Create account in OCS
          uri:
            url: "http://{{ crm_config.ocs.OCS }}/jsonrpc"
            method: POST
            body_format: json
            body:
              {
                "method": "ApierV2.SetAccount",
                "params": [{
                  "Tenant": "{{ crm_config.ocs.ocsTenant }}",

```

```

        "Account": "{{ service_uuid }}",
        "ActionPlanIds": [],
        "ExtraOptions": { "AllowNegative": false,
"Disabled": false }
    }
}
status_code: 200
register: response

- name: Add Service via API
uri:
    url: "http://localhost:5000/crm/service/"
    method: PUT
    body_format: json
    headers:
        Authorization: "Bearer {{ access_token }}"
    body:
        {
            "customer_id": "{{ customer_id }}",
            "product_id": "{{ product_id }}",
            "service_name": "Service: {{ service_uuid }}",
            "service_uuid": "{{ service_uuid }}",
            "service_status": "Active",
            "retail_cost": "{{ monthly_cost | float }}"
        }
    status_code: 200
    register: service_creation_response

- name: Add Setup Cost Transaction
uri:
    url: "http://localhost:5000/crm/transaction/"
    method: PUT
    headers:
        Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
        {
            "customer_id": {{ customer_id | int }},
            "service_id": {{
service_creation_response.json.service_id | int }},
            "title": "{{ package_name }} - Setup Costs",
            "retail_cost": "{{ setup_cost | float }}"
        }
    register: api_response_transaction

```

```
rescue:
- name: Remove account in OCS
  uri:
    url: "http://{{ crm_config.ocs.OCS }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV2.RemoveAccount",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}"
        }]
      }
    status_code: 200

- name: Fail the provision
  assert:
    that:
      - false
```

□ □ □ □ □ □ □ □ □ □

1. CRM API 接続
2. UUID 取得
3. OCS 接続
4. CRM API 接続
- 5.
6. rescue OCS 接続

--	--	--	--	--	--	--	--

OCS ActionPlans

□ □ □ □ □ □

OmniCRM ██████████

- **POST**
/crm/provision/simple_provision_addon/service_id/<id>/product_id/<id>

□□□□□□□□□□□□□□□□□□□□

- **POST**
/crm/provision/simple_provision_addon_recharge/service_id/<id>/product_id/<id>

□□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□ **IP** □□□ □ **API** □□ □□□□□□□□

- □□□□ IP □□□□□□□ □□ **crm_config.yaml** □□ **ip_whitelist** □□
- □□□□□□□□□ □□ **crm_config.yaml** □□ **api_keys** □ **API** □□
- □□□□□□□□□□□□□□

□□□ **OCS ActionPlan** □□

OCS □□□□□□□□□□□□□□□□□□□□

```
{
  "method": "ApierV1.SetActionPlan",
  "params": [{
    "Id": "ActionPlan_Service123_Monthly_Charge",
    "ActionsId": "Action_Service123_Add_Monthly_Data",
    "Timing": {
      "Years": [],
      "Months": [],
      "MonthDays": [1],
      "Time": "00:00:00Z"
    },
    "Weight": 10,
    "ActionTriggers": [
      {
        "ThresholdType": "*min_event_counter",
        "ThresholdValue": 1,
        "ActionsID": "Action_Service123_HTTP_Callback"
      }
    ]
  }]
}
```

HTTP POST

```
play_topup_no_charge.yaml
```

□ □ □ □ □ □ □ □ □

```
play_topup_monetary.yaml
```

```
- name: Mobile Topup Monetary - 2024
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: Get Product information from CRM API
      uri:
        url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_product

    - name: Get Service information from CRM API
      uri:
        url: "http://localhost:5000/crm/service/service_id/{{
service_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        register: api_response_service

    - name: Set service facts
      set_fact:
        service_uuid: "{{ api_response_service.json.service_uuid
}}"
        customer_id: "{{ api_response_service.json.customer_id }}"
        package_name: "{{ api_response_product.json.product_name
}}"
        monthly_cost: "{{ api_response_product.json.retail_cost
}}"

    - name: Get Customer Payment Method
      uri:
        url: "http://localhost:5000/api/payments/methods/default?
customer_id={{ customer_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
```



```

    return_content: yes
    register: api_response_payment_method

- name: Charge customer
  uri:
    url: "http://localhost:5000/api/payments/charge"
    method: POST
    headers:
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": "{{ customer_id | int }}",
        "amount": "{{ monthly_cost }}",
        "currency": "USD",
        "metadata": {
          "description": "{{ package_name }} topup",
          "service_id": "{{ service_id | int }}"
        }
      }
    register: api_response_payment

- name: Add monetary balance to OCS
  uri:
    url: "http://{{ crm_config.ocs.OCS }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "BalanceType": "*monetary",
          "Balance": {
            "Value": "{{ monthly_cost | float * 100 }}",
            "ExpiryTime": "+4320h"
          }
        }]
      }
    status_code: 200

- name: Add Transaction to CRM
  uri:

```


目標

OmniCRM へ API を叩き、必要なデータを取得する

必要なデータを取得する

1. 必要なデータを取得する
2. IPTV サービス ID 1 service_id
3. IP サービス ID 1 service_id

必要なデータ

- サービス ID service_id
- サービス ID extra_vars
- サービス ID Provision

実装

必要なデータを取得する

準備

必要なデータ 2

必要なデータ

- API サービス ID
- サービス ID assert: that: response.status == 200
- サービス ID
- サービス ID
- サービス ID
- OCS/OCS

必要なデータ 3

必要なデータ

- 通过 SMS/email 通知
- 配置项 `ignore_errors: True`
- 配置项 `ignore_errors: True`

配置项

配置项

- 配置项 YAML 配置
- 配置项
- 配置项
- 配置项 Ansible 配置

配置项

- Ansible 配置
- 配置 stdout 配置
- 配置 stderr 配置
- 配置 `stdout_lines` 配置
- 配置 `stderr_lines` 配置

配置项

配置项 2 配置项 `crm_config.yaml` 配置 `provisioning.failure_list` 配置

配置项

- 配置项
- 配置项/配置项
- 配置项
 - 配置项
 - 配置项
 - 配置项
- 配置项/配置项
- 配置项 Ansible 配置

API

OmniCRM APIは、CRMシステムの機能を拡張するためのAPIです。APIの仕様は、APIドキュメントを参照してください。

API

APIの仕様は、APIドキュメントを参照してください。

```
GET /crm/provision/provision_id/<id>
Authorization: Bearer <token>
```

APIの仕様は、APIドキュメントを参照してください。

```
{
  "provision_id": 123,
  "customer_id": 456,
  "customer_name": "John Smith",
  "product_id": 10,
  "provisioning_status": 0,
  "provisioning_play": "play_psim_only",
  "playbook_description": "OmniCore Service Provisioning 2024",
  "task_count": 85,
  "provisioning_result_json": [
    {
      "event_number": 1,
      "event_name": "Get Product information from CRM API",
      "provisioning_status": 1,
      "provisioning_result_json": "{...}"
    },
    {
      "event_number": 2,
      "event_name": "Create account in OCS",
      "provisioning_status": 1,
      "provisioning_result_json": "{...}"
    }
  ]
}
```

API 调用

获取设备列表

```
GET /crm/provision/?  
page=1&per_page=20&sort=provision_id&order=desc  
Authorization: Bearer <token>
```

返回结果

```
GET /crm/provision/?filters={"provisioning_status":  
[2]}&search=Mobile  
Authorization: Bearer <token>
```

返回结果中，2 代表设备状态为“Mobile”

API 说明

注意事项

- 设备 **block/rescue** 操作需要权限
- 设备 **ignore_errors** 操作需要权限
- 设备操作时 **debug** 模式需要权限
- 设备操作时 **assert** 模式 API 需要权限
- 设备操作时 **assert** 模式需要权限

其他说明

- 设备操作时需要提供 **refresh_token**
- 设备操作时 IP 地址 API 需要权限
- 设备操作时 **refresh_token** 需要权限

□□□□

- □□□□□□□□□□ customer_id□service_id □□□□□□
- □□□□□□□□□□□□□□□□□□□□□□
- □□□□□□ Provision_Event □□□□□□□□□□/□□□□

□□

- □□□□□□□□□□ customer_id□product_id□service_id
- □□□□□□□□□□□□□□□□
- □□□□□□□□action□□□□□□□□□□/□□
- **IP** □□□□□□□□□□□□□□□□□□□□

□□

- □□□□□□□□□□ API □□□□□□□□
- □□□□□□UI □□ 2-5 □□□□□□□□
- □□□□□□□□ Ansible □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□

- concepts_ansible - □□ Ansible □□□□
- concepts_sim_provisioning - SIM □□□□□□□□ eSIM□
- concepts_api - CRM API □□□□□□□□
- concepts_products_and_services - □□□□□□□□
- administration_inventory - □□□□□□□□
- □□□□□□ - □□□□□□□□□□

[illegible][illegible][illegible]

□ □

[illegible]

Product <concepts_products_and_services>

```
00000000 0000 <basics_create_customer> 00000000000000000000000000000000
000000000000000000000000
```

[illegible][illegible][illegible]

IP
<concepts_products_and_services>

第 2 部分

本部分包含以下章节：

第 2 章

第 2 章 [John ▼]

第 2 章 • John Smith ID 123 • John Doe ID 456 • Johnson Enterprises ID 789

第 2 章

- 第 2 章
- 第 2 章 ID
- 第 2 章
- 第 2 章

第 2 章

第 2 章

第 2 章

第 3 部分

第 3 部分 `inventory_items_list` 第 3 部分

□□□□□□□□

□□□□□□

SIM□ *

□□SIM□□ • SIM-00123 - ICCID□8944...0001□□□ • SIM-00124 - ICCID□
8944...0002□□□ • SIM-00125 - ICCID□8944...0003□□□

□□□□ *

□□□□□ • +44 7700 900123□□□□ • +44 7700 900124□□□□ • +44 7700
900125□□□□

□□□□□□□□

- □□□*□□□□□□□□
- □□□□□□□□□□□□“□□”□“□”□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□

- □□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□“□□”□□□“□□□”

00 4000000000

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □

[illegible]

[] []

□ □ □ □ □ □ □

- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□

[illegible]

5

[illegible]

John Smith ID 123 - 20GB

• SIM-SIM-00123 • ICCID-8944...0001 • +44 7700 900123

□□□ • □□□□£0.00 • □□□□£15.00

□□□□□□

[][] [][][][][]

□□**"□□□□□□"□□□□□□□□□□

□□ **6** □□□□□

[illegible]

✓ 手机号码 ✓ SIM卡ICCID8944...0001 ✓ 手机号码+44 7700 900123 ↻
OCS手机号码... 手机号码 手机号码

63

0.2API

☒

↻

✗

00 7000

000

000000John Smith00

00ID0789 00000000 - +44 7700 900123 000000

[00000] [00]

00**"0000"**-00000000000000

000

00000000

00000000OCS

000000000000000000000000000000

00ID0456

[00000] [00] [00]

- 000000000000000000000000
- 0000000000
- 000000000000000000000000??00

#

📍 123 Main St ▼

📧 Jane Doe - <jane@acme.com> ▼

📧 Bob Smith - <bob@acme.com> ▼

📄 PO-2025-001234_____

[📅] [📅]

📁

📁 📁 📁 📁 📁 📁 📁 📁 📁 📁

1. 📁 CSV 📁 📁 📁
2. 📁 📁 📁 📁 📁
3. 📁 📁 📁 📁 📁
4. 📁 📁 📁 📁 📁

📁

📁 📁 **1** 📁 📁 📁

1. 📁 📁 📁 📁 📁 📁
2. 📁 📁 📁 → 📁
3. 📁 📁 📁 📁 📁
4. 📁 ** "📁 - 20GB 📁" **
5. 📁 ** "📁 📁 📁" **
6. 📁 📁 📁 📁 " +1234567890 "
7. 📁 📁 **John Smith**
8. 📁 📁 📁 📁 SIM 📁
9. 📁 📁 📁 📁
10. 📁 📁 📁
11. 📁 📁 📁
12. 📁 📁 📁 📁

13. SIM卡绑定设备

任务 2 设备绑定

1. 设备名称 → **John Smith**
2. 设备类型
3. 设备**“设备”**
4. 设备类型
5. 设备**“设备 - 100Mbps”**
6. 设备类型/设备类型
7. 设备CPE设备
8. 设备类型
9. 设备
10. 设备

任务 3 设备绑定

1. 设备“设备”
2. 设备类型
3. 设备类型
4. 设备类型“设备”
5. 设备“5GB设备”
6. 设备
7. 设备类型
8. 设备类型

任务

“设备”

- 设备 设备类型
- 设备
 - 设备类型
 - 设备类型

- 認證碼驗證碼

“認證碼”

- 認證碼驗證碼
- 認證碼
 - 認證碼驗證碼
 - 認證碼驗證碼“”“”“”
 - 認證碼驗證碼驗證碼

“認證碼”

- 認證碼驗證碼
- 認證碼
 - 認證碼
 - 認證碼驗證碼ID驗證碼
 - 認證碼驗證碼

認證碼

- 認證碼驗證碼
- 認證碼
 - 認證碼驗證碼5
 - 認證碼驗證碼
 - 認證碼OCS驗證碼
 - 認證碼驗證碼

認證碼驗證碼

- 認證碼驗證碼OCS驗證碼
- 認證碼
 - 認證碼OCS驗證碼
 - 認證碼SIM/HLR/HSS驗證碼
 - 認證碼驗證碼RADIUS/DPI
 - 認證碼驗證碼

□□□□

□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□

□□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□

□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□“□□”
- □□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□

□□□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□

- `csa_add_service` - □□□□
- `concepts_products_and_services` - □□□□□□

- `administration_inventory` - 管理 inventory
- `concepts_provisioning` - 概念   provisioning
- `guide_product_lifecycle` - 产品生命周期指南



Self-Care Portal <self_care_portal>

□ □



☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

□ □ □ □ □ □

- 
- 
- 
- 

- 網路服務
- 網路設備

網路服務與網路設備

網路服務

網路服務是指透過網路提供給用戶的服務，包括以下幾種：

- 網路 - 提供用戶透過網路使用“5GB 網路”
- 網路 - 提供用戶透過網路使用“網路”
- 網路 - 提供用戶透過網路使用“WiFi 6 網路”
- 網路 - 提供用戶透過網路使用“IP 網路”

網路設備

網路設備是指用於連接網路的硬體設備，包括以下幾種：

1. 網路設備
2. 網路 “網路” 與 “網路” 網路
3. 網路設備與網路設備

網路設備

網路設備是指用於連接網路的硬體設備，包括以下幾種：

- 網路 - 提供用戶透過網路使用VoIP 網路
- 網路 - 提供用戶透過網路使用網路
- 網路 - 提供用戶透過網路使用網路
- 網路 - 提供用戶透過網路使用網路

網路設備與網路設備

網路設備

網路設備是指用於連接網路的硬體設備，包括以下幾種：

Modem-12345 - TP-Link AX1800 • Modem-12346 - TP-Link AX1800 • Modem-12347 - Netgear RAX40

3

[] []

-
-

4

“”

✓ ✓ ↺ ...

-
- Ansible (provisioning_play)
-
-
- OCS
-

Complete Product Lifecycle Guide - Adding Addons
<guide_product_lifecycle>

項目別料金

月額料金（基本料金） “基本” 月額料金

オプション

5GB 追加

2025年1月10日 から 2025年1月17日 まで 月額 £5.00

WiFi 6 対応機器

2024年12月1日 から Modem-12345 まで 月額 £10.00

注意事項

重要事項

- 本サービスは、
- “基本” 月額料金
- オプション “5GB” の 月額料金
- 適用

詳細情報

- サービス - 月額料金
- サービス - 月額料金
- サービス - 月額料金
- サービス - 月額料金

注

本サービスは、CPE（Customer Premises Equipment）として提供されます。

お問い合わせ

お問い合わせ 1

- お問い合わせ “お問い合わせ”
- お問い合わせ

3. 選擇“**???**”
4. 選擇“5GB ”
5. 選擇
6. 選擇

第 2 步 選擇 IP

1. 選擇 VPN 的 IP
2. 選擇
3. 選擇“IP ”
4. 選擇 IP
5. 選擇
6. 選擇 IP

第 3 步 選擇

1. 選擇
2. 選擇
3. 選擇“WiFi 6 ”
4. 選擇
5. 選擇
6. 選擇 £10.00

選擇

“ ”

- 選擇
- 選擇 `service_type` 為 `residential/business`

選擇

- 選擇
- 選擇
 - 選擇
 - 選擇

- 管理する製品を登録

管理する製品を登録

- 製品を登録する
- 製品の詳細を登録
 - 製品の詳細を登録
 - 製品の詳細を登録
 - 製品の詳細を登録 `inventory_items_list` を登録

管理する製品

- 製品を登録する
- 製品の詳細を登録
 - 製品の詳細を登録 `auto_renew: true`
 - 製品の詳細を登録
 - 製品の詳細を登録

管理する製品

- `guide_product_lifecycle` - 製品の詳細を登録
- `csa_add_service` - 製品の詳細を登録
- `concepts_products_and_services` - 製品の詳細を登録
- `administration_inventory` - 製品の詳細を登録

--	--	--	--

[illegible]

```

<csa_add_service>
<csa_service_usage>
<csa_modify>
<concepts_products_and_services>

```

--	--	--	--	--	--

□□□□

□□□□

□ □

--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

11

- **ID** - 00000
- 0000 - 0000000000“00 - +44 7700 900123”0

- 網路 - 網路服務品質
- 網路 - 網路IPTV網路VoIP
- 網路 - 網路服務品質
- 網路 - 網路服務
- 網路 - 網路/網路
- 網路 - 網路
- 網路 - 網路服務品質

網路

網路

網路服務品質

- 網路 - 網路服務品質
- 網路 - 網路服務品質
- 網路 - 網路服務品質
- 網路 - 網路服務品質

網路服務品質

網路

網路服務品質

網路服務品質

網路

- 網路 - 網路 SIM 網路
- 網路 - 網路DSL 網路
- **IPTV** - 網路服務品質
- **VoIP** - 網路服務品質

□□□□

□□□□□□□□

□□□□□□□□□□□□□□

□□□□□□□□□□□□

□□ □□□□□□□□□□□□

□□

□□□□□□□□□□□□□□

□□□

- □□□□
- □□ ID
- □□□□
- □□□□□□□□□□□□
- □□ UUID

□□ □□“0770”□□□□□□□□□□□□“0770”□□□□

□□□□

□□□□□□□□□□□□□□

□□□□□

- □□ ID□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□

□□□□□

- 00000000A-Z000-00000-000
- 00000000Z-A000-00000-000
- 00000000 ▲ 0 ▼ 000

□ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □

1. 00 - 0000000000000000
2. 00 - 00000000SIM 0000000000
3. 00 - 0000000000
4. 0000 - 00/00/SMS 0000
5. 0000 - 000000000000
6. 0000 - 0000

□ □ □ □

□□□□□□□□□□□□□□ : □□□□□□□□□□

- 00 - 000000
- 01 - 000000
- 0000 - 000000000000
- 0000 - 000000000000
- 00 - 00/0000

☒ 1
☒ 2
☒ 3

- 1.
2. “”
- 3.

-
-
-
-

111

- 
- 
- 

[illegible]

--	--	--	--	--	--

□ □ □ □

□ □ □ □ □ □ □ □ □ □

11

- 
- 
- 
- 

111

- 00000000
- 0000
- 00000000
- 0000000
- 00000000/00

11

- 00000
- 0000000000000000
- 0000000
- 00000/0000
- 0000000000000000000000000000000000

❓❓ 問 題

問 題

1. 問 題
2. 問 “ ”
3. 問 “ ”
4. 問 “ ”

問 題

- 問 → 問 題
- 問 → 問 題 OCS/
- 問 → 問 題
- 問 → 問 題

問 題

問 “ ”

問 題

- 問 - 問 題
- 問 題 - 問 題
- 問 題 - 問 題

- 0000 - 0000/00000000

111

- -

--	--	--	--	--	--	--	--	--

[illegible]

--	--	--	--	--	--

□□□□

□□□□

--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

11

- □□ - □□□□□□□□□□

5/5

API 仕様

CGRateS からの API 呼び出し

- CGRateS から ActionPlan を取得
- CGRateS から ActionPlan を更新
- CGRateS から ActionPlan を削除
- CGRateS から ActionPlan を作成

API

CGRateS から `ApierV1.GetScheduledActions` API を CRM 経由で呼び出す

API

API の呼び出し

- API の呼び出し (API) の
- API の呼び出し

Acme Corp から FixedWireless_75628fa5 から Mega から \$89.99

2025年11月11日 10:45

[API] [API]

- API “API” の

4.

5.

-
- CGRateS
-
-
-

-
-
-
-

1. ()

2.

Acme Corp FixedWireless_75628fa5 20251111 10:45

[] []

3. “

4. ActionPlan CGRateS

5.

□□□□□□

- ActionPlan → CGRateS →
- →
- →
- →

111

- 
- 
- 
- 

11

□ □

--	--	--	--	--	--

□□□□

- □□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□□□□□□□□□“N/A”

□□□□

- 000000/UUID
- 0000000000
- 0000000000000000 UUID
- 000000000000000000000000 "N/A"

□□□□

- CGRateS ActionPlan

- 000000000000
- 000000000000000000

0000

- 00 CRM 0000000000
- 0000000000
- 0000000000000000 “N/A”

000000

- 0000 00000000
- 000000000000 “3000” “200”
- 00000000 CGRateS 000000

0000

- 00 CRM 0000000000
- 000000 - 000000
- 000000 - 000000
- 000000 - 0000000000000000

0000

00000/000000 “N/A”

- 000 CGRateS 0000 UUID 0 CRM 00000 service_uuid 000
- 000
 - 0000000000000000
 - 00 service_uuid 00000000 ServiceType_UUID 00000
FixedWireless_75628fa5 0
 - 0000000 CRM 00000000 CGRateS 000

0000 “N/A”

- 000 CGRateS 0000 ID 0 CRM 00000000
- 000
 - 0000000000000000

- 通过 ActionPlan 通过 product_id 通过 CRM 通过
- 通过

“通过”通过

- 通过 CGRateS 通过 NextRunTime
- 通过
 - 通过 CGRateS ActionPlan 通过
 - 通过 ActionTiming 通过
 - 通过 CGRateS 通过

“通过”通过

- 通过 通过
- 通过
 - 通过
 - 通过 Ansible 通过
 - 通过 OCS 通过
 - 通过

通过

- 通过 通过 CGRateS 通过
- 通过
 - 通过 “通过” 通过
 - 通过 ActionPlan 通过 CGRateS 通过
 - 通过 CGRateS API 通过

通过

通过

◀ 通过 1 [2] 3 4 5 通过 ▶

通过[10 ▼]

- 10

--	--	--	--	--	--	--	--	--

CGRateS Ansible

- CGRateS 00000000
- 000000IMSI00MSISDN0000000000000000
- 0000000000000000
- 000000000000

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- `MaxBitrateDL/UL` - 最大下行/上行速率
- `PcefPolicyName` - QoS 策略名称
- `MSISDN` - 手机号码
- - 设备名称
- - 设备类型

[illegible]

- □□□□ - □□□□□□□□□□□□
- □□ - □□□□□□□□
- □□□□ - □□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□□□□□□□□□

11

CGRateS

CGRateS

CGRateS

- **CGRateS** 000 - 0000000000
- **CGRateS** 0000 - 00000000000
- 00000 - 00000000000

[illegible]



CGRateS 簡介

CGRateS 是一個用於記錄和處理呼叫記錄的系統。

它的 ID 是 ATTR_ACCOUNT_{service_uuid}

它的 UUID 是 Mobile_SIM_c2880638 它的 ID 是 ATTR_ACCOUNT_Mobile_SIM_c2880638

它的

1. 記錄呼叫記錄
2. 它的 “CGRateS 簡介” 簡介
3. 它的 “CGRateS 簡介” 簡介
4. 它的 000000/00/0000
5. 它的 “CGRateS 簡介”

□□□□□

- □□ - □□□□□□□□□□ *req.Account□ *req.IMSI□
- □□ - □□□□□□□□

- `*constant` - 常數
- `*variable` - 由 RSRParser 解析的變數
- `*composed` - 組成的規則
- `*usage_difference` - 使用差異
- `*sum` - 總和
- `*value_exponent` - 值指數
- 其他 - 其他規則

JSON 格式

```
{
  "Path": "*req.Account",
  "Type": "*constant",
  "Value": [{"Rules": "Mobile_SIM_474a380a"}]}
}
```

規則

- 規則名稱
- IMSI/MSISDN 規則
- MaxBitrateDL/UL
- PcefPolicyName
- 其他規則

規則

規則名稱

CGRateS 規則

規則名稱

規則 ID 規則 `FLTR_ACCOUNT_{service_uuid}`

規則 規則 UUID 規則 `Mobile_SIM_c2880638` 規則 ID 規則 `FLTR_ACCOUNT_Mobile_SIM_c2880638`

測試步驟

1. 準備測試環境
2. 將 “CGRateS 測試” 填
3. 將 “測試環境” 填
4. 將/將/將
5. 將 “測試環境”

測試結果

- 將 - 測試環境測試 ~*req.Account ~*req.Destination
- 將 - 測試環境
 - *string - 測試環境
 - *prefix - 測試環境
 - *suffix - 測試環境
 - *empty - 測試環境
 - *exists - 測試環境
 - *notexists - 測試環境
 - *timings - 測試環境/測試環境
 - *destinations - 測試環境
 - *rsr - RSR 測試環境
 - *gt / *gte / *lt / *lte - 測試環境
- 將 - 測試環境測試環境測試環境

測試環境

```
{
  "Element": "~*req.Account",
  "Type": "*string",
  "Values": ["Mobile_SIM_474a380a"]
}
```

測試結果

- 測試環境測試環境
- 測試環境測試環境

- □ □ □ □ □ □ □ □ □ □

--	--	--	--	--	--	--

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒ ☒ ☐

□ □ □ □

[illegible]

□ □ □ □ □ □ □

5. 如何“管理”你的时间

□□□□□□□□

- □□□□ - □□□□□□
- □□□□ - □□□□□□□□□□□□□□
- □□□ - □□□□□□□□□□

□□□□□□□□

□□“□□□□”□□□□□□□□□□□□

- 項目名
 - CGRID項目 ID
 - 項目名 UUID
 - 項目名
 - 項目名/項目名
 - 項目
 - 項目
- 項目名項目
 - 項目名 JSON 項目
 - 項目 CGRateS 項目
 - 項目名項目
 - 項目名 JSON 項目名項目

□□□

- □□□□□□□□□□
- □□□□□□□□

- 000000000000
- 0000000000
- 000000000000

0000

00000000000000000000“00”000000000000

00

000000000000 UUID000000000000 `*string:~*req.Account:{service_uuid}` 0000
00

CGRateS API 00

00 CGRateS 0000000000000000 OmniCRM API 00000

000 `POST /crm/ocs/proxy`

00000

- `method` - CGRateS API 000000 `APIerSv1.GetAttributeProfile`
- `params` - 00000000
- `customer_id` - 00 ID000000000000
- `service_id` - 00 ID000000000000

00000

- `tenant` - CGRateS 000000000000

00000

```
{
  "method": "APIerSv1.GetAttributeProfile",
  "params": [{"ID": "ATTR_ACCOUNT_Mobile_SIM_c2880638"}],
  "customer_id": 123,
  "service_id": 456
}
```


目標

- 利用可能な CGRateS API を知る
- 利用可能な
 - 利用可能な API の一覧を知る
 - API の JSON レスポンスを知る
 - 利用可能な API の一覧を知る
 - API の CGRateS を知る

利用可能な API

- 利用可能な API の一覧を知る
- 利用可能な
 - 利用可能な API の一覧を知る
 - 利用可能な API の一覧を知る
 - 利用可能な UUID を API の CGRateS を知る

利用可能な API

- 利用可能な API の一覧を知る
- 利用可能な API の一覧を知る

利用可能な CGRateS API

- 利用可能な GET API の一覧を知る
- 利用可能な API - API/利用可能な API

利用可能な API

利用可能な 1 利用可能な API

1. API の利用可能な API
2. API の利用可能な API
3. API の利用可能な API
4. API の利用可能な API
5. API の利用可能な API

問題 2 問題文

1. 文 “ ”
2. 文 “ ”
- 3.
- 4.

問題 3 問題文

1. 文 “ ”
- 2.
- 3.
4. 文 “ ”
- 5.

問題 4 問題文

- 1.
- 2.
- 3.
4. 文 “ ”
- 5.

問題 5 問題文

1. 文 “ ”
- 2.
3. 文 “ ”
4. / / SMS
- 5.

- 0000 000000000000
- 0000 00“00000”0000000000

000000000000

- 0000 000000000000
- 0000
 - 000000000000“0770”0000000000
 - 0000000000
 - 000000000000

000000000000

- 0000 0000000000
- 0000 0000000000000000000000

000000

- 0000 00000000
- 0000 000000 ▲▼ 000000000000

00000000

- 0000 000000000000
- 0000
 - 0000000000000000
 - 000000000000
 - 000000000000

000000

- `csa_add_service` - 00000000
- `csa_modify` - 0000000000000000
- `concepts_products_and_services` - 0000000000
- `basics_customers` - 000000

자세한 정보

자세한 정보에 대한 자세한 정보는 OCS Self-Care Portal <self_care_portal>에서 확인하실 수 있습니다.

주요

주요 사항

- 사용자 - 사용자 정보
- 사용자 - 사용자 정보
- 사용자 - 사용자 정보
- 사용자 - 사용자 정보
- 사용자 - 3 사용자 정보

자세한 정보

자세한 정보

1. 사용자 정보 → 사용자
2. 사용자 정보 사용자 (:)
3. 사용자 "사용자"

자세한 정보

1. 사용자 정보
2. 사용자 "사용자" 사용자

자세한 정보

1. 사용자 정보
2. 사용자 정보 사용자
3. 사용자 정보 "사용자"

자세한 정보에 대한 자세한 정보는   사용자

□□□□□□

□□□□□□□□□□□□□□□□□□□□

□□□

□□□□□□□□□□□□□□□□

□□□□□

- 記憶體 - 記憶體容量/記憶體類型
- 記憶體 - 記憶體容量/記憶體類型
- 記憶體 - 記憶體容量/記憶體類型
- 記憶體 - 記憶體容量/記憶體類型

記憶體

記憶體容量/記憶體類型

記憶體

- 記憶體 - 記憶體容量/記憶體類型
- 記憶體 - 記憶體容量/記憶體類型
- 記憶體 - 記憶體容量/記憶體類型“12.5GB / 20GB”
- 記憶體
 - 記憶體>50% 記憶體
 - 記憶體20-50% 記憶體
 - 記憶體<20% 記憶體
- 記憶體 - 記憶體容量/記憶體類型

記憶體

記憶體

記憶體容量/記憶體類型

記憶體 記憶體 (GB) 記憶體 (MB)

記憶體

記憶體12.5GB / 20GB (62%)

記憶體

- 記憶體 - 5GB/10GB/20GB 記憶體
- 記憶體 - 記憶體容量/記憶體類型500GB/1TB

- $10^2 - 10^8$
- $10^4 - 10^9$

□□□□

□ □ □ □ □ □ □ □ □ □ □

- 
- 
- 

□ □ □ □

□□□□□□□□

□□□ □□ (min)

□□□□□

125 min / 500 min (25%)

□□□□□□□

- 
- 
- 
- 

111

□ □

□ □ □ □

□□□□□□□□

000 00 (msgs)

□□□□□

45 / 250 (18%)

Features

- 160 GB
-
-

Price

Price per GB

Price per GB (£, \$, €, ¥)

Price per GB

£15.50 / £20.00 (77%)

Features

-
-
-
-

3

"" "

Price

20GB
12.5 GB 7.5 GB 2025 1 25 15 10

5GB
5.0 GB 0 GB 2025 1 31 21 20

0 GB 3.0 GB 2025 1 18 30

□ □ □ □ □

- **ID/** -
- -
- -
- -
- -
- -

□ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □

- $\square\square\square = \square\square\square\square$
- $\square\square\square = \square\square\square\square$

□ □ □ □ □

- □□□□□□ 30□□□□□□□□□□
- □□□□□□ 20□□□□□□
- □□□□□□ 10□□□□□□□□□□

[illegible]

CGRateS CGRateS Actions and Topup Behaviors

□ □ □ □

□□□□□ **3** □ □□□□□□□

□ □ □ □ □

- ☐ ☐ ☐ ☐ ☐ ☐

- 1000000
- 1000000
- 1000000000

1000000

- 1000000000
- 1000000000
- 1000000000
- 1000000000000000

1000

- 1000000000000000
- 1000000000000000000
- 100000000000

1000000000000000000

100000

1000000000000000

- 100 (GB)
- 100 (min)
- 100 (msgs)
- 100 (pounds)

1000

100012.5GB 100 1000125 min 100 100045 msgsg 100 1000£15.50 100

100000/1000

1000000

- 空间 (GB 或 TB)
- 网络带宽

其他

450GB / 500GB 每月 45.00 英镑

网络/带宽

网络带宽

- 网络带宽
- 网络延迟

其他

`dongle=true` 需要硬件设备

其他

网络带宽 0 / 0

- 网络带宽 OCS 或 CGRateS 支持
- 网络
 - 网络带宽 OCS 支持
 - 网络 OCS API 支持
 - 网络 UUID 支持

其他

- 网络带宽 OCS 支持
- 网络
 - 网络
 - 网络带宽
 - 网络 OCS API 支持

其他


```
{
  "BalanceMap": {
    "*data": [
      {
        "ID": "monthly_data_20GB",
        "Value": 13421772800,
        "ExpiryTime": "2025-01-25T23:59:59Z",
        "Weight": 10
      }
    ],
    "*voice": [
      {
        "ID": "monthly_voice_500min",
        "Value": 7500000000000,
        "ExpiryTime": "2025-01-25T23:59:59Z",
        "Weight": 10
      }
    ],
    "*sms": [
      {
        "ID": "monthly_sms_250",
        "Value": 250,
        "ExpiryTime": "2025-01-25T23:59:59Z",
        "Weight": 10
      }
    ],
    "*monetary": [
      {
        "ID": "prepaid_credit",
        "Value": 1550,
        "ExpiryTime": "2025-02-25T23:59:59Z",
        "Weight": 10
      }
    ]
  }
}
```

5. UI 1111111111111111 → GB1111 → 1111

6. □□□□□□□□

7. 3 00000

OCS □□□□□□

OCS 認證服務與認證服務 UI 說明

- `*data` → 數據 認證服務
- `*voice` → 語音 認證服務
- `*sms` → 簡訊 認證服務
- `*monetary` → 貨幣 認證服務

認證服務與認證服務 + 認證 + 認證服務與認證服務??

認證服務

- 認證 認證 → GB 認證 1024^3
- 認證 認證 → 認證 60×10^9
- 認證 認證服務
- 認證 認證 → 認證 100

認證服務與認證服務

認證服務與認證服務 OCS 認證服務與認證服務與認證服務與認證服務與認證服務與認證服務 **CGRateS Actions and Topup Behaviors**

認證服務與認證服務

認證服務與認證服務 CGRateS 認證服務與認證服務與認證服務與認證服務與認證服務與認證服務

- 認證服務與認證服務
- 認證服務
- 認證服務
- 認證服務

認證服務與認證服務

1. 認證服務

- 認證服務 `auto_renew = true`
- 認證服務 OCS 認證服務
- 認證服務與認證服務與認證服務與認證服務與認證服務與認證服務

2. 設定項目

設定項目

- 設定 **ID** - 設定 UUID
- 設定 - 設定時間間隔（時間間隔）
- 設定 - 設定時間間隔（時間間隔）1 日 00:00 UTC
- 設定 - 設定時間

3. 設定項目

- OCS 設定項目
- 設定項目（設定項目）20GB 設定 500 設定項目
- 設定項目（設定項目）30 設定項目
- 設定項目（設定項目）

4. 設定項目

- 設定 OCS 設定項目
- 設定項目（設定項目）
- 設定項目（設定項目）

設定項目

```
{
  "ActionPlanId":
  "ProductID_MonthlyPlan__ProductName_20GB_Mobile__ActionPlan_Monthly_F
  "NextExecTime": "2025-02-01T00:00:00+00:00",
  "ActionName_hr": "設定項目",
  "PlanName": "20GB 設定項目",
  "ActionFrequency_hr": "設定項目",
  "custom_NextExecTime_hr": "設定 22 設定項目"
}
```

設定項目

- 設定 - 設定項目（設定項目）
- 設定 - 設定 OCS 設定項目（設定項目）
- 設定 - 設定項目（設定項目）

□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- 
- 

UI 1111

□ □ □ □ □ □ □ □ □ □ □ □

2025年2月22日 星期五 £15.00

4444

□ □ □ □ □ □ □

- □□□“□□□□□□□□”□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□

□ □ □ □ □ □ □ □ □ □

- 
- 
- 
- 

□ □ □ □ □ □

- 8 bits
- 16 bits
- 32 bits OCS 16 bits
- 32 bits

--	--	--	--

- **CGRateS Actions and Topup Behaviors** - 中国移动充值业务
- **CGRateS Destinations** - 中国移动 PLMN 充值业务
- **features_topup_recharge** - 充值业务
- **csa_service_management** - 充值业务
- **csa_modify** - 充值业务
- **concepts_products_and_services** - 充值业务

--	--	--	--

[illegible]

Self-Care Portal <self_care_portal>

RBAC <rbac> Customers <basics_customers> Self-Care Portal <self_care_portal>

11

□ □ □ □ □ □ □

1. $1000 - 1000000000000$
2. $10 - 1000000000000$
3. $10 - 1000000000$
4. $10 - 1000000000000000000000$
5. $1000 - 1000000000000000000$

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- `can_impersonate` — 16 bit
- `admin` — 16 bit

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

📦📦📦📦📦

📦📦 **Web UI**📦

1. 📦📦📦📦 — 📦 CRM 📦📦📦📦
2. 📦📦📦📦 — 📦📦📦📦📦📦📦📦
3. 📦📦“📦📦📦📦📦📦” — 📦📦📦📦📦📦📦📦📦📦📦📦📦📦
4. 📦📦📦📦 — 📦📦📦📦📦📦📦📦
5. 📦📦📦📦 — 📦📦📦📦📦📦📦📦📦📦

📦📦 **API**📦

📦📦📦📦📦📦

📦📦📦 `POST /auth/impersonate`

📦📦📦📦 `can_impersonate` 📦 `admin`

📦📦📦

```
{
  "user_id": 42
}
```

JSON

```
{
  "success": true,
  "impersonating_user_id": 1,
  "target_user_id": 42,
  "impersonation_start": "2025-01-04T15:30:00Z",
  "access_token": "new_token_for_impersonated_user",
  "refresh_token": "new_refresh_token"
}
```

JSON 格式

JSON 格式

JSON 格式

- 响应 — 成功时返回的 JSON 对象
 - `success`
 - `impersonating_user_id`
 - `target_user_id`
 - `impersonation_start`
- 请求 — 请求体中的 JSON 对象
 - `impersonating_user_id` — impersonating 用户 ID
 - `target_user_id` — target 用户 ID
 - `impersonation_start` — impersonation 开始时间
- 消息 — UI 显示的消息
 - "成功 [用户] 登录"
 - "失败"
- 日志 — 记录 impersonation 的日志
 - 记录成功
 - 记录失败
 - 在 `ImpersonationLog` 中记录

API

Web UI

1. ログイン — ログイン画面
2. ログアウト — ログアウト画面“ログアウト”
3. プロフィール — プロフィール画面

API

POST /auth/stop_impersonation

成功時レスポンス

失敗時レスポンス

```
{
  "success": true,
  "impersonation_end": "2025-01-04T15:45:00Z",
  "duration_seconds": 900,
  "access_token": "your_original_token",
  "refresh_token": "your_original_refresh_token"
}
```

失敗時レスポンス

API

成功時レスポンス ImpersonationLog

- **impersonating_user_id** — ログインユーザーID
- **target_user_id** — impersonation 対象ユーザーID
- **impersonation_start** — impersonation 開始時刻
- **impersonation_end** — impersonation 終了時刻
- **impersonation_duration** — impersonation 継続時間

失敗時レスポンス

- 管理員 — 管理員權限
- 管理員 — 管理員權限
- 管理員 — 管理員權限
- 管理員 — 管理員權限

管理員權限

管理員權限 API 管理員權限

GET /auth/impersonation_logs?user_id={user_id}&start_date={date}&end_date={date}

管理員權限

- 管理員權限
- 管理員權限
- 管理員
- 管理員

管理員

1. 管理員 — 管理員權限
2. 管理員 — 管理員權限
3. 管理員 — 管理員權限
4. 管理員 — 管理員權限
5. 管理員 — 管理員權限
6. 管理員 — 管理員權限 `can_impersonate`
7. 管理員 — 管理員權限

管理員

- 管理員 — 管理員權限
- 管理員 — 管理員權限
- 管理員 — 管理員權限

- 密碼 — 密碼驗證碼驗證碼驗證碼驗證碼
- 密碼 — 密碼驗證碼驗證碼
- **MFA** 碼 — 密碼 2FA 密碼驗證碼驗證碼

密碼

- 密碼驗證碼 — 密碼驗證碼驗證碼
- 密碼 — 密碼驗證碼驗證碼驗證碼
- 密碼 — 密碼驗證碼驗證碼
- 密碼 — 密碼驗證碼

密碼驗證碼

密碼“密碼驗證碼”密碼

- 密碼驗證碼 `can_impersonate` 或 `admin` 碼
- 密碼驗證碼驗證碼

密碼驗證碼

- 密碼驗證碼驗證碼
- 密碼驗證碼

密碼驗證碼

- 密碼驗證碼 API 密碼驗證碼 `POST /auth/stop_impersonation`
- 密碼驗證碼 cookies 密碼驗證碼

密碼驗證碼

- 密碼驗證碼驗證碼
- 密碼驗證碼 - 密碼驗證碼 ID

📖📖📖 CGRateS 📖📖

📖📖📖📖📖 OmniCRM 📖📖📖📖📖📖 CGRateS 📖📖📖

📖📖📖 CGRateS 📖📖📖 Python 📖📖📖📖📖📖📖📖📖 CRM 📖 OCS📖📖📖📖📖📖

📖📖

📖📖📖📖📖📖📖📖📖📖📖📖

1. **CGRateS/OCS** - 📖📖📖📖📖📖📖📖📖📖📖
2. **OmniCRM** - 📖📖📖📖📖📖📖📖📖📖📖

📖📖📖📖📖📖📖📖📖📖📖📖 Ansible Play 📖 CGRateS 📖📖📖

📖📖📖📖CGRateS 📖📖

CGRateS 📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖 OmniCRM 📖📖📖 CGRateS 📖📖📖📖📖

- 📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖
- 📖📖📖📖📖📖📖 MO 📖 MT 📖📖/SMS 📖📖📖📖📖📖📖📖
- 📖📖📖📖📖📖 2G/3G 📖📖 CAMEL📖MO-MT SMS📖USSD📖📖📖📖📖📖📖📖 LTE/5G 📖 Diameter
- 📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖
- 📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖LCR📖📖📖📖📖📖

📖📖 **CRM** 📖📖 - 📖📖📖📖 OmniCRM 📖📖📖📖📖📖📖📖📖📖📖📖📖📖📖 CGRateS 📖📖📖📖📖
CGRateS 📖📖📖📖 CGRateS 📖📖📖📖📖📖📖📖📖📖📖

Python Requests

第 1 部分 CRM API 认证

```
import requests
import json

crm_url = 'https://crm.example.com/'
session = requests.Session()

headers = {
    "Content-Type": "application/json"
}

# 登录
response = session.post(crm_url + '/crm/auth/login', json={
    "email": "admin@example.com",
    "password": "your_password"
}, headers=headers)

assert response.status_code == 200
headers['Authorization'] = 'Bearer ' + response.json()['token']
print("CRM 认证成功")
```

第 2 部分 数据获取

使用认证后的 session 获取 CRM 数据


```

inventory_list_new = []

# SIM
inventory_list_new.append({
    "item": "SIM Card",
    "itemtext1_label": "ICCID",
    "itemtext2_label": "IMSI",
    "wholesale_cost": 0.2,
    "retail_cost": 1,
    "allow_dropdown_staff": True,
    "allow_dropdown_customer": True,
    "icon": "fa-solid fa-sim-card"
})

# 
inventory_list_new.append({
    "item": "Mobile Number",
    "itemtext1_label": "E.164 Mobile Number",
    "itemtext2_label": "Type",
    "wholesale_cost": 0.2,
    "retail_cost": 3,
    "icon": "fa-solid fa-phone"
})

# 
for inventory_template in inventory_list_new:
    # 
    inventory_list_existing = session.get(
        crm_url + '/crm/inventory/item_template/',
        headers=headers
    )

    # 
    if inventory_template['item'] in [x['item'] for x in
inventory_list_existing.json()]:
        # 
        inventory_template_id = [
            x['inventory_template_id']
            for x in inventory_list_existing.json()
            if x['item'] == inventory_template['item']
        ][0]

        response = session.patch(

```

```

        crm_url + '/crm/inventory/item_template/' +
str(inventory_template_id),
        json=inventory_template,
        headers=headers
    )
    assert response.status_code == 200
    print(f"成功: {inventory_template['item']}")
else:
    # 失败
    response = session.put(
        crm_url + '/crm/inventory/item_template',
        json=inventory_template,
        headers=headers
    )
    assert response.status_code == 200
    print(f"成功: {inventory_template['item']}")

```

返回数据

- **item** - 名称
- **itemtext1_label** - 第一行文本
- **itemtext2_label** - 第二行文本
- **wholesale_cost** - 批发成本
- **retail_cost** - 零售成本
- **allow_dropdown_staff** - 是否允许下拉选择员工
- **allow_dropdown_customer** - 是否允许下拉选择客户
- **icon** - Font Awesome 图标

3. CGRateS

```

import cgrateshttpapi
import time

OCS_Obj = cgrateshttpapi.CGRateS("ocs.example.com", "2080")

tenant = "your_tenant_name"
tpid = str(tenant) + "_" + str(int(time.time()))

```

第4章 CGRateS の実装

この章では、CGRateS の実装について説明します。

実装の流れ

1. 初期化 - 設定ファイル/DB から `Dest_AU_Mobile`、`Dest_International_UK` を読み込む
2. **PLMN** の取得 - 設定ファイル/DB から `Dest_PLMN_OnNet`、`Dest_PLMN_US_Verizon` を読み込む

処理の流れ

- 設定ファイル/DB から `Dest_AU_Mobile`、`Dest_International_UK` を読み込む
- 設定ファイル/DB から `Dest_PLMN_OnNet`、`Dest_PLMN_US_Verizon` を読み込む

処理の流れ

- 設定ファイル/DB から `Dest_AU_Mobile`、`Dest_International_UK` を読み込む
- `PLMN` の取得
- `PLMN` の取得 `mccXXX.mncYYY`
- 設定ファイル/DB から `Dest_PLMN_OnNet`、`Dest_PLMN_US_Verizon` を読み込む

設定ファイル `CGRateS` の内容 - 設定ファイル

第5章 CGRateS の実装

この章では、CGRateS の実装について説明します。

この章では、CGRateS の実装について説明します。

設定ファイル `CGRateS` の内容 - 設定ファイル

- 設定ファイル `Python` の内容 `OCS_Obj.SendData`
- 設定ファイル `*topup`、`*topup_reset` の内容
- 設定ファイル `SMS` の内容
- 設定ファイル `BalanceId`、`BalanceType`、`DestinationIDs`、`Units`、`ExpiryTime`、`Weight`
- 設定ファイル `BalanceId`、`BalanceType`、`DestinationIDs`、`Units`、`ExpiryTime`、`Weight`
- 設定ファイル `BalanceId`、`BalanceType`、`DestinationIDs`、`Units`、`ExpiryTime`、`Weight`

□□□□□□

- ***topup** - 00000000000000000000000000000000
- ***topup_reset** - 00000000000000000000000000000000

6 CRM

CGRateS CRM

□□□□□ **SIM** □□

```
product_list_new = []

product_list_new.append({
    "category": "standalone",
    "provisioning_play": "play_psim_only",
    "relies_on_list": "",
    "contract_days": 0,
    "retail_cost": 0,
    "retail_setup_cost": 0,
    "product_slug": "Mobile-SIM",
    "product_name": "SIM",
    "comment": "SIM",
    "provisioning_json_vars": '{"iccid" : "", "msisdn" : ""}',
    "terms": "6 12",
    "service_type": "mobile",
    "residential": True,
    "business": True,
    "enabled": True,
    "inventory_items_list": "['SIM Card', 'Mobile Number']",
    "icon": "fa-solid fa-sim-card",
    "features_list": "['(61xxx)', '', '']",
    "wholesale_cost": 3,
    "wholesale_setup_cost": 1,
    "customer_can_purchase": True
})
```

□ □ □ □ □ □ □ □

```

product_list_new.append({
    "category": "addon",
    "provisioning_play": "play_topup_charge_then_action",
    "relies_on_list": "",
    "contract_days": 30,
    "retail_cost": 59.00,
    "retail_setup_cost": 0,
    "product_slug": "au-premium-plan-1", # Action_au-premium-
plan-1
    "product_name": "AU 100GB 3000 1",
    "comment": "100GB 3000 3000 6GB SMS",
    "provisioning_json_vars": "",
    "terms": "30",
    "service_type": "mobile",
    "residential": True,
    "business": True,
    "enabled": True,
    "icon": "fa-solid fa-mobile",
    "features_list": "100GB 3000 3000 SMS 6GB",
    "wholesale_cost": 45.00,
    "wholesale_setup_cost": 0,
    "auto_renew": "True",
    "customer_can_purchase": True
})

```

100GB 3000 3000 6GB

```

product_list_new.append({
    "category": "addon",
    "provisioning_play": "play_topup_charge_then_action",
    "relies_on_list": "",
    "contract_days": 0,
    "retail_cost": 30,
    "retail_setup_cost": 0,
    "product_slug": "au-data-addon-20gb", # 名称 Action_au-data-
addon-20gb
    "product_name": "AU 数据 20GB",
    "comment": "数据 20GB 包",
    "provisioning_json_vars": "",
    "terms": "数据服务条款",
    "service_type": "mobile",
    "residential": True,
    "business": True,
    "enabled": True,
    "icon": "fa-solid fa-mobile",
    "features_list": "['数据 20GB 包']",
    "wholesale_cost": 22,
    "wholesale_setup_cost": 0,
    "auto_renew": "False",
    "customer_can_purchase": True
})

```

数据服务

- **category**名称 - 名称
 - standalone - 数据服务
 - addon - 数据服务
 - bundle - 数据服务
- **provisioning_play**名称 - 名称 Ansible playbook名称
 - play_topup_charge_then_action play_simple_service
- **relies_on_list** - 名称 product_slugs 名称
- **contract_days** - 名称 0 = 名称 30 = 名称
- **retail_cost**名称 - 名称/名称
- **retail_setup_cost** - 名称
- **product_slug**名称 - 名称 **CGRates** 名称 Action_{product_slug}
- **product_name**名称 - 名称

- **comment** - 備註
- **provisioning_json_vars** - 在 playbook 中定義的 JSON 變數
- **terms** - 條款
- **service_type** - 服務類型
- **residential** - 住宅
- **business** - 商業
- **enabled** - 是否啟用
- **inventory_items_list** - 在 Python 中定義的列表
- **icon** - Font Awesome 圖示
- **features_list** - 在 Python 中定義的列表
- **wholesale_cost** - 批發成本
- **wholesale_setup_cost** - 批發設置成本
- **auto_renew** - 是否自動續訂 "True" 或 "False"
- **customer_can_purchase** - 客戶是否可以購買

📄📄📄📄📄

7 API

```
# 
product_list_existing = session.get(crm_url + '/crm/product',
headers=headers)
existing_products = product_list_existing.json()

# 
for product in product_list_new:
    print(f": {product['product_slug']}")

    # 
    if product['product_slug'] in [x['product_slug'] for x in
existing_products]:
        print(f": {product['product_slug']}")

        # product_id
        product_id = [
            x['product_id']
            for x in existing_products
            if x['product_slug'] == product['product_slug']
        ][0]

        product['product_id'] = product_id

    # 
    response = session.patch(
        crm_url + '/crm/product/' + str(product_id),
        json=product,
        headers=headers
    )
    print(f": {response.status_code}")
    assert response.status_code == 200
    print(f": {product['product_slug']}")
else:
    # 
    print(f": {product['product_slug']}")
    response = session.put(
        crm_url + '/crm/product',
        json=product,
        headers=headers
    )
    print(f": {response.status_code}")
```



```
assert response.status_code == 200
print(f"response: {product['product_slug']}")
```

Playbooks

CRM API CGRateS API

```
# CGRateS
ActionsId = "Action_au-premium-plan-1"

# CRM
product_slug = "au-premium-plan-1"

# playbook
cgr_action_name = "Action_" + product_slug
# : "Action_au-premium-plan-1"
```

Playbooks

ExecuteAction playbook SetBalance
CGRateS API OmniCRM-
API/Provisioners/plays/play_topup_charge_then_action.yaml
ExecuteAction Playbooks

product_slug

```
# play_topup_charge_then_action.yaml
- name: cgr_action_name
  set_fact:
    cgr_action_name: "Action_{{
api_response_product.json.product_slug }}"

#
- name: CGRateS
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "APIerSv1.ExecuteAction",
        "params": [{
          "Tenant": "{{ crm_config.ocs.ocsTenant }}",
          "Account": "{{ service_uuid }}",
          "ActionsId": "{{ cgr_action_name }}"
        }]
      }
}
```

CGRateS " "

Ansible Playbooks

"50GB "

1. CGRateS

5 50GB

- **ActionsId:** "Action_50gb-data-pack" product_slug
- **BalanceType:** "*data"
- **DestinationIDs:** "Dest_PLMN_OnNet"
- **Units:** 50 * 1024 * 1024 * 1024 50GB

- **Identifier:** `"*topup"` `*****`

2. `CRM` `*****`

```
product_50gb_data = {
    "category": "addon",
    "provisioning_play": "play_topup_charge_then_action",
    "relies_on_list": "",
    "contract_days": 0,
    "retail_cost": 25.00,
    "retail_setup_cost": 0,
    "product_slug": "50gb-data-pack", # ActionsId *****
    "Action_"
    "product_name": "50GB *****",
    "comment": "***** 30 ***** 50GB *****",
    "provisioning_json_vars": "",
    "terms": "***** 30 *****",
    "service_type": "mobile",
    "residential": True,
    "business": True,
    "enabled": True,
    "icon": "fa-solid fa-database",
    "features_list": "['50GB *****', '30 *****', '*****']",
    "wholesale_cost": 20.00,
    "wholesale_setup_cost": 0,
    "auto_renew": "False",
    "customer_can_purchase": True
}

# CRM *****
response = session.put(
    crm_url + '/crm/product',
    json=product_50gb_data,
    headers=headers
)
assert response.status_code == 200
print("*****: 50gb-data-pack")
```

3. `*****`

`*****`

1. 创建 "50GB 套餐" 产品 slug: 50gb-data-pack
2. Playbook 创建 play_topup_charge_then_action
3. Playbook 创建 Action_50gb-data-pack
4. Playbook 调用 APIerSv1.ExecuteAction 执行 ActionsId Action_50gb-data-pack
5. CGRateS 创建套餐
6. 创建 50GB 套餐

Playbooks

--	--	--	--	--	--

CRM CGRateS CGRateS

1

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ SMS ☐ ☐

CRM ☐☐☐

```
{
  "product_slug": "premium-monthly-plan", # 🏷️ Action_premium-
monthly-plan
  "product_name": "📶📶📶",
  "category": "addon",
  "retail_cost": 59.00,
  "auto_renew": "True",
  "contract_days": 30,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": "['100GB 📶', '5000 📶', '5000 📶 SMS', '📶
📶']"
}
```

```

topup_reset 1 Python

```

步骤 2 创建 10GB 流量包

创建 10GB 流量包

CRM 配置

```
{
  "product_slug": "10gb-addon", # 名称 Action_10gb-addon
  "product_name": "10GB 流量包",
  "category": "addon",
  "retail_cost": 15.00,
  "auto_renew": "False", # 是否自动续费
  "contract_days": 0,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": "['10GB 流量包', '30 天有效']"
}
```

创建 10GB 流量包 *topup 套餐 CGRateS 套餐 - 流量包

步骤 3 创建 50GB 固定套餐

创建 50GB 固定套餐

CRM 配置

```
{
  "product_slug": "fixed-50gb-monthly", # 名称 Action_fixed-50gb-monthly
  "product_name": "50GB 固定套餐",
  "category": "addon",
  "retail_cost": 35.00,
  "auto_renew": "True",
  "contract_days": 30,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": "['50GB 固定', '流量包', '30 天有效']"
}
```

创建 50GB 固定套餐 *topup_reset 套餐 CGRateS 套餐 - 流量包

Table of Contents

Table of Contents CGRateS Table of Contents

- Table of Contents MO & MT & SMS & USSD
- Table of Contents 2G/3G & CAMEL & MO-MT SMS & USSD & 4G/5G & Diameter
- Table of Contents Table of Contents
-  Table of Contents

Table of Contents OmniCRM Table of Contents PAYG Table

Table of Contents

Table of Contents SMS Table

1. CGRateS Table of Contents GB
2. Table of Contents
3. Table of Contents CGRateS Table of Contents
4. Table of Contents \$0.10 & MB
5. Table of Contents

API 调用示例

1. 设置TP目的地费率

```
OCS_obj.SendData({
    "method": "ApierV2.SetTPDestinationRate",
    "params": [{
        "TPid": tpid,
        "ID": "DR_DATA_PAYG",
        "DestinationRates": [{
            "DestinationId": "Dest_PLMN_OnNet",
            "RateId": "RT_DATA_$0_10_PER_MB",
            "RoundingMethod": "*up",
            "RoundingDecimals": 4,
            "MaxCost": 0,
            "MaxCostStrategy": ""
        }]
    }]
})
```

2. 设置TP费率

```
OCS_obj.SendData({
    "method": "ApierV2.SetTPRate",
    "params": [{
        "TPid": tpid,
        "ID": "RT_DATA_$0_10_PER_MB",
        "RateSlots": [{
            "ConnectFee": 0,           # 连接费
            "Rate": 0.10,             # 费率 $0.10
            "RateUnit": "1048576",    # 1 MB * 1024 * 1024 字节
            "RateIncrement": "1048576", # 1 MB 增量
            "GroupIntervalStart": "0s"
        }]
    }]
})
```

3. 设置TP计费计划

```
OCS_obj.SendData({
    "method": "ApierV2.SetTPRatingPlan",
    "params": [{
        "TPid": tpid,
        "ID": "RP_PAYG_DATA",
        "RatingPlanBindings": [{
            "DestinationRatesId": "DR_DATA_PAYG",

```

```

        "TimingId": "*any",
        "Weight": 10
    }]
}]
})

# 4. 設定TPRatingProfile
OCS_Obj.SendData({
    "method": "ApierV2.SetTPRatingProfile",
    "params": [{
        "TPid": tpid,
        "Tenant": tenant,
        "Category": "data",
        "Subject": "*any", # 設定Subject
        "RatingPlanActivations": [{
            "ActivationTime": "2024-01-01T00:00:00Z",
            "RatingPlanId": "RP_PAYG_DATA",
            "FallbackSubjects": ""
        }]
    }]
})

# 5. 設定TariffPlan
OCS_Obj.SendData({
    "method": "APIerSv1.LoadTariffPlanFromStorDb",
    "params": [{
        "TPid": tpid,
        "DryRun": False,
        "Validate": True,
        "APIOpts": {}
    }]
})

```

注意事項

- 設定TPRatingProfile時，**\$0.10** 為 **MB**
- 設定TPRatingProfile時，`*monetary` 為
- 設定TPRatingProfile時，**1 MB** 為

--	--	--	--	--	--	--	--	--

```
# 0000000000000000
OCS_Obj.SendData({
    "method": "ApierV2.SetTPRate",
    "params": [{
        "TPid": tpid,
        "ID": "RT_VOICE_DOMESTIC_$0_30_PER_MIN",
        "RateSlots": [{
            "ConnectFee": 0.15,          # $0.15 0000
            "Rate": 0.30,                # 000 $0.30
            "RateUnit": "60s",          # 000000
            "RateIncrement": "60s",     # 00000000
            "GroupIntervalStart": "0s"
        }]
    }]
})

OCS_Obj.SendData({
    "method": "ApierV2.SetTPRate",
    "params": [{
        "TPid": tpid,
        "ID": "RT_VOICE_INTL_$1_50_PER_MIN",
        "RateSlots": [{
            "ConnectFee": 0.25,          # $0.25 0000
            "Rate": 1.50,                # 000 $1.50
            "RateUnit": "60s",
            "RateIncrement": "1s",       # 00000000
            "GroupIntervalStart": "0s"
        }]
    }]
})
```

□ □ □ □ □ □ □ □ □ □

[illegible]

```
OCS_Obj.SendData({
    "method": "APIerSv1.SetChargerProfile",
    "params": [{
        "Tenant": tenant,
        "ID": "Charger_Default",
        "FilterIDs": [],
        "AttributeIDs": ["*constant:*req.RequestType:*none"],
        "Weight": 999
    }]
})
```

□□□□□

- CGRateS 00000000 - 000000000000000000000000
- CGRateS 0000 - 00000000 PLMN 0000

--	--	--	--

1.

```
# ④ - ④④④④④④
ActionsId = "Action_premium-plan-100gb"
product_slug = "premium-plan-100gb"

# ④ - ④④④④④④④④
ActionsId = "Action_Plan_A"
product_slug = "premium_100"
```

2. □□□□□□□□

CRM CGRateS playbook

3. **CDR**

□□□□ *cdrlog □□□□□□□□□□

```
{
  "Identifier": "*cdrlog",
  "BalanceType": "*generic",
  "ExtraParameters": "
{\"Category\\": \"^activation\\", \"Destination\\": \"Product Name\\"}",
  "Weight": 80
}
```

4. 平衡ID

```
# 100GB
"BalanceId": "Premium_Data_100GB"
"BalanceId": "AU_Voice_Domestic__" + str(units)

# data1
"BalanceId": "data1"
"BalanceId": "balance"
```

5. 平衡单位

```
# 100GB - 100GB
Units = 100 * 1024 * 1024 * 1024 # 100GB 100GB
Units = 3000 * 60 * 1000000000 # 3000 100GB

# 107374182400
Units = 107374182400
```

6. 平衡单位

平衡单位 = 平衡单位 * 平衡单位

```
# 重置账户 = 700
{"Identifier": "*reset_account", "Weight": 700}
{"Identifier": "*topup_reset", "Weight": 90}
{"Identifier": "*cdrlog", "Weight": 80}

# 充值 = 1000
"BalanceWeight": 1200 # 充值/重置
"BalanceWeight": 1100 # 充值
"BalanceWeight": 1000 # 充值
```

7. 充值重置

充值重置

```
# 充值
Units = 100 * 1024 * 1024 # 100MB 充值

# 重置
Units = 100 * 1024 * 1024 * 1024 # 100GB
```

充值

充值重置

充值 Playbook 充值 "充值" 在 "ActionsId 充值"

充值 充值 CGRateS 充值

充值

1. 充值 CGRateS 充值
2. 充值 `ActionsId = "Action_" + product_slug`
3. 充值

返回“充值成功”

返回 充值成功消息

返回 DestinationIDs 充值成功消息

返回

1. 返回 CGRateS 充值成功
2. 返回充值成功 *any
3. 返回 ID 充值

充值成功消息

返回 充值成功消息

返回 充值成功消息

返回

1. 返回 *topup 或 *topup_reset 充值
2. 返回充值成功 0
3. 返回充值成功

充值成功

返回 充值成功消息

返回 返回 *topup_reset 充值 *topup

返回

- 返回 Identifier 或 *topup_reset 充值 *topup
- 返回 Balanceld 充值成功

返回 CGRateS 充值成功

□□

□□◆◆□□□□ OmniCRM □□□□ CGRateS □□□□□□□□□□□□□□□□□□□□
□MVNO□□□□□□□□/□□□□□□□□□□□□

□□□□ CGRateS □□□□□□□□□□□□□□□□□□□□□□□□

- □□□□ SMS □□□□ MO/MT □□
- □□ 2G/3G □□□□□□ CAMEL □□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□/□□/□□□
- □□□□□□ QoS □□□
- □□□□□□□□□□□□

...□□□□□ CGRateS □□□□□□ CRM □□□□□□□□□□□□

□□□□

- **CGRateS □□□□□□** - □□□□□□□□□□□□□□□□
- **Ansible Playbooks □□** - Playbooks □□□□□□
- **□ Playbooks □□□□□□** - □□□□□□□□□□
- **□□□□□□** - □□□□□□□□□□
- **□□□□□□□□** - □□□□□□□
- **CGRateS □□□□** - □□□ CGRateS □□



OmniCRM 系統與 AMBER 系統整合

系統 SMS 功能與系統整合

OmniTouch 系統與 OWL 系統

OmniTouch 系統與 OWL 系統整合

- 系統與 **CBE** - 系統與 Web 系統
- 系統與 **CBC** - 系統與 2G/3G/4G/5G 系統

OWL 系統與 Web 系統整合 Chrome/Firefox/Safari/Edge 系統



系統與 PWS 系統

- 系統 - 系統
- 系統 - 系統
- 系統 - 系統 500 系統
- 系統 - 系統
- 系統 - 系統 CBC 系統
- 系統 - 系統 TOTP 系統
- 系統 - 系統

系統 SMS 系統與系統整合 10 系統

11

□ □ □ □ □ □ □

- 2010 - 2011
- **AMBER** - 2012
- 2013 - 2014
- 2015 - 2016
- 2017 - 2018

[illegible]

□ □

□ □ □ □ □ □ □ □ □ □ □ □

1. -
2. -
3. -
4. -

□ □

1

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □

[illegible]

☐ OWL ☐ MI

1111

[illegible]

500

“”

-
-
-
-
-

-
-

2

OWL

-
-
-

- □ □ □ □ □ □ □ □ □ □

□□4□□□

[illegible][illegible]

□□5□□□□□

□ □

□□□□

[illegible]

□ □ □ □ □ □

[illegible]

--	--	--	--

□ □ □ □ □ □ □ □ □

- □ □ □ □ □ □ □ □ □ □

- □ □ □ □ □ □ □ □ □

- /

--	--	--	--

[illegible]

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□

- 000000 - 00000000000000000000ETWS00004370ETWS00004371

- □□ - □□□□□□□□□□□□□□□□□□□□

- □□□□ - □□□□□□□□

- 品牌 - 品牌名称
- 品牌 - 品牌名称
- 品牌 - 品牌名称

品牌

品牌CBC品牌名称

- 品牌 - ISO品牌名称en-es-fr-zh
- 品牌 - 品牌名称1395品牌

品牌名称品牌名称品牌名称品牌名称品牌名称

品牌

品牌名称品牌名称

- 品牌 - 品牌名称ID品牌名称
- 品牌 - 品牌名称MCC-MNC
- **RAT** - 品牌名称LTE5GUMTS GSM

品牌名称品牌名称品牌名称品牌名称品牌名称

品牌名称品牌名称

品牌Web UI

1. 品牌名称 - 品牌名称CBC品牌名称
2. 品牌“品牌名称” - 品牌名称品牌名称
3. 品牌名称
 - 品牌名称品牌名称4370
 - 品牌名称品牌名称
 - 品牌名称品牌名称5-60
 - 品牌名称999品牌名称品牌名称
 - 品牌名称品牌名称品牌名称
 - 品牌名称品牌名称“品牌”

4. 國際化支援

- 支援“國際化”
- 支援國際化支援
- 支援國際化GSM7位元1395位元Unicode支援
- 支援國際化支援

5. 國際化支援

- 支援“國際化支援”
- 支援國際化支援
- 支援國際化MCC-MNC支援
- 支援RAT支援LTE 5G支援
- 支援國際化支援

6. 000000 - 0000000000“0000”

00**API**0

000 PUT /crm/cbc/

00000 CREATE_CBC_MESSAGE

00000

```
{
  "messageIdentifier": "4370",
  "category": "emergency",
  "repetitionPeriod": 10,
  "numberOfBroadcasts": 999,
  "warningPeriodSec": 3600,
  "channelIndicator": "basic",
  "localized_messages": [
    {
      "language": "en",
      "messageBody": "EARTHQUAKE WARNING: Magnitude 6.5 earthquake
detected. Take cover immediately. Drop, Cover, Hold On."
    },
    {
      "language": "es",
      "messageBody": "ADVERTENCIA DE TERREMOTO: Terremoto de
magnitud 6.5 detectado. Cúbrase inmediatamente. Agáchese, Cúbrase,
Agárrese."
    }
  ],
  "tracking_areas": [
    {
      "tracking_area": "12345",
      "operator": "310-410",
      "rat_type": "LTE"
    },
    {
      "tracking_area": "12346",
      "operator": "310-410",
      "rat_type": "5G"
    }
  ]
}
```

□□□

```
{
  "cbc_message_id": 123,
  "cbc_unique_id": "550e8400-e29b-41d4-a716-446655440000",
  "messageIdentifier": "4370",
  "category": "emergency",
  "repetitionPeriod": 10,
  "numberOfBroadcasts": 999,
  "warningPeriodSec": 3600,
  "channelIndicator": "basic",
  "initiating_user": 5,
  "approving_user": null,
  "created": "2025-01-10T14:30:00Z",
  "localized_messages": [...],
  "tracking_areas": [...]
}
```

□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□

□□□ GET /crm/cbc/

□□□□□ VIEW_CBC_MESSAGE

□□□□ CBC□□□□□□□□□□□□□□□□□□□

□□□□

□□□ PATCH /crm/cbc/{cbc_message_id}

□□□□□ UPDATE_CBC_MESSAGE

□□□□□□□□□□□□□□□□□□□□□□□□ CBC□

□□□□□


```
{
  "cbc_message_id": 123,
  "numberOfBroadcasts": 500,
  "localized_messages": [
    {
      "language": "en",
      "messageBody": "UPDATED: Earthquake warning still in effect.
Aftershocks possible."
    }
  ]
}
```

--	--	--	--

```
DELETE /crm/cbc/{cbc_message_id}
```

DELETE_CBC_MESSAGE

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ CBC ☐ ☐ ☐ ☐

--	--	--	--	--	--

□ □

1. 送信 - 送信元アドレス `initiating_user`
2. 送信 - 送信元アドレス `approving_user`

□ □ □ □ □ □ □

- approving_user = null
- “”
-
- approving_user

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ ? ? ? □

--	--	--	--	--

3GPP TS 23.041

ETWS

- **4370** - ETWS□□□□
- **4371** - ETWS□□□□
- **4372** - ETWS□□□□□□□□
- **4373-4378** - ETWS□□□□□□
- **4379** - ETWS□□□□

CMAS/WEA

- **4352** - □□□□
- **4353-4355** - □□□□
- **4356-4359** - □□□□
- **4360-4363** - AMBER□□
- **4364-4367** - □□□□□□
- **4368-4369** - □/□□□□
- **4380-4381** - □□□□

□□□□□□

- **0-999** - □□□□□□□□
- **1000-4095** - □□□□□□

☐ CBE ☐ OWL ☐ CBC

□□□□□

[illegible]

OmniCRM CBC

OmniCRM CBC OWL CBC

OWL CBC

CBC URL `crm_config.yaml`

```
cbc_url: "http://cbc.example.com:8080"
```

CBC

1. OmniCRM CBE
2. CBC API
3. `{cbc_url}/alerts/send` HTTP POST
4. CBC CBSP SBc-AP N50
5. CBC
- 6.

CBE CBC

2G/3G/4G/5G /

OWL 2G 3G 4G 5G

CBSP - 2G/3G - BSC

CBSP CBC 2G GSM BSC

- 2G 3G /
- BSC

- 3GPP RNC BSC OWL CBC
- 3GPP RNC BSC OWL CBC

3GPP RNC BSC OWL CBC SABP RNC/BSC CBSP RNC

SBc-AP - 4G/5G - MME/IWF

SBc-AP OWL CBC 4G/5G eNodeB/gNodeBs MME

- 4G LTE
- 5G 2025
- MME OWL CBC
- SBc-AP

N50 - 5G - AMF

5G N50 OWL CBC 5G gNodeBs AMF

- OWL CBC
- 2025 5G SA AMF
- 5G SA

MNO

OWL CBC

- CBC MNO
-
- MME RNC BSC

OWL CBC

OWL❖❖

- **NetAct**
- **U2000 / U2020**
- **NetNumen / ZXPOS**
- **ENM**

Omnitouch

RBAC

OWL RBAC

RBAC

- - OWL
- -
- -
- -

RBAC

1. -
2. -
3. -

CRUD

- -
- -
- - ❖❖

- 消息 - 消息内容

消息CBC

- CREATE_CBC_MESSAGE - 创建消息
- VIEW_CBC_MESSAGE - 查看消息
- UPDATE_CBC_MESSAGE - 更新消息
- DELETE_CBC_MESSAGE - 删除消息

消息内容

消息内容 <rbac>

消息内容

消息

消息内容

1. 消息内容URL
2. 消息内容
3. 消息内容2FA消息内容2FA消息

消息内容

消息内容 <authentication_flows>

消息

消息内容

1. 消息内容“消息”
2. 消息内容
3. 消息

消息内容

消息内容 → 消息内容

1. 消息内容 → 消息

□□□□

□□□□ `VIEW_CBC_MESSAGE` □ `CREATE_CBC_MESSAGE`

□□□□

□□□□

□□□□

- □□□□ - □□□□□□□□□□
- □□□□□ - □□□□□□□□□□□□
- □□□□ - □□□□□□□□
- □□□□ - □ `cellSites.txt` □□□□

□□□□

- □□□□□ - □□□□□□□□□□
- □□□□ - □□□□□□□□□□
- □□□□ - □□□□□□□□□□

□□□□

- □□□□ - □□□□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□□□

□□□□

□□**1**□□□□□□□□

□□□□□□□□□□□□□□□□□□□□ `/cellSites.txt` □□□□□□□□

- **lat/lng** - □□□□
- **site_name** - □□□
- **sectors** - □□□□□□□□□□□□

□□**2**□□□□□□□□

目標設定

1. 目標設定の重要性

- 方向性を定める
- モチベーションを高める
- 進捗を確認できる

2. 目標設定の方法

- SMART原則
- 短期目標と長期目標
- 目標達成のための計画

3. 目標達成のモチベーション

- 自己肯定感
- 目標へのコミットメント
- 周囲のサポート

第3章 目標設定のツール

目標設定ツール

- 目標設定シート
- 目標管理システム
- 目標設定アプリ
- 目標設定カード

目標管理

第4章 目標達成のモチベーション

モチベーションの向上

- 自己肯定感の向上
- 目標へのコミットメント
- 周囲のサポート
- 目標達成の喜び

目標

目標を達成するための手順

手順

1. **CBC**のインストール
2. データの取得
3. データの加工
4. データの可視化
5. **CBC**のインストール

目標を達成するための手順

目標

目標

目標を達成するための手順

- データ - データ/データ
- データ - データ
- データ - データ/データ/AMBER
- データ - データ/データ

目標を達成するための手順

目標 `site_data.json`

```
[
  {
    "area": "Coastal Region",
    "polygon": [
      {
        "coord": [
          {"lat": -33.8688, "lng": 151.2093},
          {"lat": -33.8650, "lng": 151.2070},
          {"lat": -33.8600, "lng": 151.2150},
          {"lat": -33.8688, "lng": 151.2093}
        ]
      }
    ]
  }
]
```

□□□□□□□□

1. □□□□□□□□□□
2. □□□□□□□□□□
3. □□□□□□□□□□□□
4. □□□□□□□□□□

□□□□

Google Maps API

□□□□□□□□□□Google Maps API□□□

```
REACT_APP_G00GLE_API_KEY=your_google_maps_api_key_here
```

□□□□□

□□□□□□□□□□`public/cellSites.txt`□

```
# lat,lng,name,sector1,sector2,sector3
-33.8688,151.2093,SYD_CENTRAL,0,120,240
-33.8650,151.2070,SYD_HARBOUR,45,165,285
```

1111111111

- 000000
- 000000
- 00GPS00000
- 00000000RNC000

□□□□□

mapStyles.js

- 
- 
- 
- 

11

□□□□□

- 
- 
- 
- 

555

- 
- 
- 
- 

55

- 國際標準
- 國際標準
- 國際標準國際標準
- 國際標準

國際標準

國際標準國際標準Omnitouch國際標準國際標準OWL國際標準

國際標準

- 國際標準國際標準國際標準360國際標準
- 國際標準國際標準國際標準國際標準
- 國際標準國際標準“國際標準”“國際標準”
- 國際標準
- 國際標準國際標準國際標準

國際標準

- 國際標準國際標準國際標準
- 國際標準國際標準國際標準
- 國際標準國際標準國際標準
- 國際標準國際標準Unicode國際標準

國際標準

- 國際標準國際標準國際標準國際標準
- 國際標準國際標準國際標準國際標準
- 國際標準國際標準國際標準國際標準
- 國際標準國際標準國際標準國際標準

國際標準

- 國際標準國際標準國際標準4379國際標準4380-4381國際標準
- 國際標準國際標準
- 國際標準CBC國際標準國際標準

- □ □ □ □ □ □ □ □ □

□□□□□

- 
- 
- 
- 

--	--	--	--	--	--

2FA□□□□□

- 2FA
-
- 2FA
- 2FA
- 2FA

2FA <2fa>

□ □ □ □

□□□□□□□□

Omnitouch NDMO

GIS Google Earth Omnitouch

□□□□□□□□

NDMO

- 
- 
- 

Microsoft Windows [Version 6.0.6002.18005] Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\>cd C:\Program Files\Microsoft Windows Defender\

- C:\Program Files\Microsoft Windows Defender\
- C:\Program Files\Microsoft Windows Defender\
- C:\Program Files\Microsoft Windows Defender\
- C:\Program Files\Microsoft Windows Defender\

C:\>

C:\>cd C:\Program Files\Microsoft Windows Defender\

- C:\Program Files\Microsoft Windows Defender\
- C:\Program Files\Microsoft Windows Defender\
- C:\Program Files\Microsoft Windows Defender\

C:\>

CBE / NDMO /

3

- 2 CBE
- 1

C:\>

- 50GB
- CPU 2 CPU
- RAM 8GB
- Omnitouch
- TCP 443 CBC TLS

CBCMNO

3

- 2CBC
- 1

- 50GB
- CPU 2CPU
- RAM 8GB
- Omnitouch
- MME/RNC/BSC
- CBCTCP443CBC

OWL

1. -
2. -
3. CBC - CBC
4. / -
5. -
6. IP - NDMO MNO IP
7. VPN - Omnitouch
8. CBC NDMO -
9. CBC MNO -
10. - CBC
11. CBC CBC -
12. CBC - BSC/MME/AMF
13. API NMS -
14. -
15. / -

16. 00000000 - 0000000000

0000

00000000000000000000

GSM 70000000000000000

- 00093000
- 000150 × 93 = 001395000

Unicode UCS-20000000000000000

- 00041000
- 000150 × 41 = 00615000

OWL000

- 00000000500000
- Web UI00000000000000000000

000000

0000000000000000

- 00000000000000
- 00000000
- 0000000000
- CBC API000000
- 00000000

000000000000000000

```
SELECT * FROM cbc
WHERE created >= '2025-01-01'
ORDER BY created DESC;
```

□□□□□□□□

□□□□□□□□□□□□□□

□□□□

- □□□2015□□□□□□□□□□□□□□
- □□□□□□□□□
- □□□□□□□□□□□□/SMS□□□
- □□□□□□□□□SMS□□□□□□□

□□□□

- □□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□
- □□□□/□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□

1. □□□□□□□□□□4379□4380-4381□
2. □□□□□□□□□□□□
3. □□□□□□□□□□□□□□□
4. □□□□□□□□□□□□□□□

□□□□

OWL□□□□□□□□□□□□□□□□□□□□

□□□□□/SMS

□□□□□□□SMS□□□□□□□□

- 00000000000000000000000000000000
- 00000000000000000000000000000000
- 0000000000000000

00000000

00000000000000000000

- 000000
- 0000000000000000
- 0000000000
- 0000000000000000

0000000000

00000000000000000000

- 00000000000000
- 00000000000000
- 00000000000000

000000/00

000000000000000000

- 000000
- 0000000000000000
- 000000000000
- 00000000000000

0000000

OWL CBC000000000000

- 000000
- 00API

- 000000
- 0000
- 00000000
- 0000

0000000000000000

0000000

0000

0000000000000000

- 000000000000
- 000000000000000000000000
- 0000000
- 000000000000

00000

- 00000437904380-438100000000
- 0000000000000000000000
- 0000000000000000
- 000000000000

0000000

◆◆ 00000000000000000000/00000000Omnitouch00000000000000000000

000000

OWL000000000000

- 000NetAct
- 00U2000 / U2020
- 00NetNumen / ZXPOS

- **ENM**

□ □ □ □ □ □

OmniTouch

[illegible]

--	--	--	--	--	--	--

[illegible]

NDMO

GIS Google Earth Omnitouch

--	--	--	--	--	--	--

□□□□ □□□□□□□□□□□□NDMO□□□□□□

- 
- 
- 
- 

OmniTouch

□ □ □ □ □ □

□ □ □ □ □ □

[illegible]

- 
- 
- 

- □ □ □ □ □ □ □ □ □ □

2FA / □□□□

Omnitouch□□□□□□□□□□

- 2FA
- /
-

□ □ □ □ □ □ □

[illegible]

□ □ □ □ □ □ □ □ □ □

1. □□□□□□□□□□
2. □□□□□□□
3. □□□□□□
4. □□□□□□□□□□

API

API

API

- 00/00 - 0000000000000000
- 0000 - 000000000000
- 0000 - 00000000000000000000
- 0000 - 0000000000000000

□□□□□API□□□□□□□□□□□□□□□□?□

API

CBC

POST

PUT /crm/cbc/

GET

GET /crm/cbc/

PUT

PATCH /crm/cbc/{cbc_message_id}

DELETE

DELETE /crm/cbc/{cbc_message_id}

API Swagger /crm/docs/

OmniCRM

OmniCRM 是用于管理客户关系的系统

```
Customers <basics_customers>
Inventory
<administration_inventory>
Service Management
<csa_service_management>
```

CRM 系统

CRM 系统

系统功能

系统功能

数据

数据

数据

1. 数据 - 数据
2. 数据 - 数据
3. 数据 - 数据
4. 数据 - 数据 ICCID 数据 itemtext1 itemtext2
5. 数据 - 数据 UUID

数据

- 数据 - 数据 "Smith" "John Smith" "Smithson"
- 数据 - "john" "John" "JOHN" "john"
- 数据 - 数据
- 数据 - 数据 10 数据

□□□□

□□□□

1. □□□□□□□□□□
2. □□**“□□”**□□□□

□□□□□□

- □□□□ "Acme Corp"
- □□□□ "+1234567890" □ "1234567890"
- □□□□ "john@example.com" □ "john"
- □□□□ "ICCID8944" □□ "8944"
- □□ UUID □ "123e4567-e89b"

□□□□□□□□

□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

1. □□**“□□□□□□□□”**□□□□
2. □□□□**“□□”**

□□□□□□

- □□□ customer_status = "Closed" □□□
- □□□□□□□□□□□□□□□□□□

Information

- Information
- Information
- Information
- Information

Information

Information

Information

John Smith ID: 123 role: customer

John Smith (role) ID: 123 role: contact

Mobile - +44 7700 900123 ID: 123 role: service

Information

- [redacted] - Information
- ID - Information
- [redacted] - Information

Information

Information

Information

Information

Information

Information

Information

- 電話番号 - “Smith” “John Smith” 電話番号 ❶❷❸
- 名前 - “John” 電話番号
- 電話番号 - 電話番号
- 電話番号 - 電話番号
- 名前 **UUID** - 電話番号

電話番号

名前

- 電話番号

名前

- 名前
- 名前
- 電話番号
- 電話番号

名前

- 電話番号

名前

- itemtext1 電話番号 ICCID 電話番号 MAC 名前
- itemtext2 電話番号 IMSI 電話番号
- *電話番号 itemtext3-20 電話番号*

名前

- 電話番号
- 名前 UUID

電話番号

- 電話番号
- 電話番号
- 電話番号

- 項目名
- 項目ID
- 項目タイプ
- 項目テキスト itemtext1/2

API 例

検索

```
GET /utilities/search_everything?
search=Smith&page=1&per_page=10&search_closed_records=false
Authorization: Bearer <token>
```

レスポンス

- search項目 - 検索結果
- page項目 - ページ番号1
- per_page項目 - ページサイズ10
- search_closed_records項目 - 検索結果を閉じるfalse

例


```
{
  "data": [
    {
      "id": 123,
      "name": "John Smith",
      "customer_id": 123,
      "type": "customer"
    },
    {
      "id": 456,
      "name": "John Smith",
      "customer_id": 123,
      "type": "contact"
    },
    {
      "id": 789,
      "name": "Mobile - +44 7700 900123",
      "customer_id": 123,
      "type": "service"
    }
  ],
  "pagination": {
    "current_page": 1,
    "per_page": 10,
    "total_pages": 5,
    "total_items": 47
  }
}
```

□□□□□□□□

□□□□□□□□□□ SQL UNION□

```

-- Customers
SELECT customer_id AS id,
       customer_name AS name,
       customer_id,
       'customer' AS type
FROM customer
WHERE customer_name LIKE '%Smith%'
      AND customer_status = 'Open'

UNION ALL

-- Contacts
SELECT contact_id AS id,
       CONCAT(contact_firstname, ' ', contact_lastname) AS name,
       customer_id,
       'contact' AS type
FROM customer_contact
WHERE (contact_firstname LIKE '%Smith%' OR
      contact_lastname LIKE '%Smith%' OR
      contact_email LIKE '%Smith%' OR
      contact_phone LIKE '%Smith%')

UNION ALL

-- Sites
SELECT site_id AS id,
       site_name AS name,
       customer_id,
       'site' AS type
FROM customer_site
WHERE site_name LIKE '%Smith%'

UNION ALL

-- Inventory
SELECT inventory_id AS id,
       itemtext1 AS name,
       customer_id,
       'inventory' AS type
FROM inventory
WHERE itemtext1 LIKE '%Smith%' OR
      itemtext2 LIKE '%Smith%'

```

UNION ALL

```
-- Services
SELECT service_id AS id,
       service_name AS name,
       customer_id,
       'service' AS type
FROM customer_service
WHERE service_name LIKE '%Smith%' OR
       service_uuid LIKE '%Smith%'
```

□□□□□□□□□□

□□□□

□□□□□

- □□□□□□□□□□ LIKE □□□ %term%□
- □□□□□□□□□□
- □□□□□□>100k □□□□□□□□□□□□□□□□
- □□□□□□□□ 10 □□□□□□□

□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□

□□□□□□□□□□□□

- □□□ □□□□□□□□□□□□□□□□
- □□□
 - □□□□□“+44 7700 900123”□□□“07700900123”□□□□
 - □□□□□□□□“<john.smith@example.com>”□□□“john”□□□□

--	--	--	--	--	--	--

OmniCRM Self-Care Portal

```
<self_care_portal> 00000000000000000000000000000000
```

- 物联网 - 物联网 SIM 物联网卡
- 物联网 - WiFi 模块 dongles 物联网模块
- 物联网 - 物联网设备

SIM Card Provisioning <concepts_sim_provisioning>

SIM

11

Stripe

□□□□□

- 000000000000000000
- 0000000000001-30 00
- 000000000000
- Stripe 0000000000
- 000000000000
- 00000000
- 000000000000

--	--	--	--	--	--

□□□□□□ □ **URL** □□□□□□□□□□ CRM □□□□□□

□ □ □ □ □ □ □

- 000000 SMS 000000
- 0000000000

- 國際漫遊服務
- 國際漫遊

國際漫遊服務 IP 地址 IMSI 國際漫遊服務

國際漫遊

國際漫遊 **4** 國際漫遊

1 國際漫遊

國際漫遊服務國際漫遊服務

國際漫遊

- 國際漫遊 - 國際漫遊 1 至 30 天
- 國際漫遊 - 國際漫遊服務
- 國際漫遊 - 國際漫遊服務
- 國際漫遊 - 國際漫遊服務/國際漫遊

國際漫遊

國際漫遊

- 國際漫遊服務 `REACT_APP_TOPUP_PRICE_PER_DAY` 國際漫遊
- 國際漫遊 \$10 國際漫遊
- 國際漫遊 `REACT_APP_CURRENCY_CODE` 國際漫遊

2 國際漫遊

國際漫遊服務國際漫遊

- 國際漫遊
- 國際漫遊
- 國際漫遊

國際漫遊

- 
- 
- 
- 

3

Stripe Elements PayPal SDK

--	--	--	--	--	--	--	--

- ☐ ☐ ☐ Visa ☐ Mastercard ☐ Amex ☐
- ☐ ☐ ☐
- ☐ ☐ ☐ ☐ Apple Pay ☐ Google Pay ☐ PayPal ☐ ☐ ☐ ☐ ☐ ☒ ☒

□□□□□

- PCI 100000000000
- OmniCRM 100000000000
- 3D Secure 100000000000
- 100000000000

□□□□□

1.
2.
3.
4.
5.

11

[illegible]

4

□□□□□

2025 年 1 月 17 日

客户ID: <customer@example.com> 交易ID: TXN-123456

尊敬的客户:

感谢您使用我们的服务。

您的订单已成功处理。

订单详情如下:

商品名称: 无线网卡

订单号:

订单金额: 120.00 元

1. 支付信息

支付方式:

- 支付状态: succeeded
- 支付金额: days × price_per_day
- 支付时间: 2025-01-17 10:00:00

2. 服务信息

- API 接口: POST /oam/topup_dongle
- 服务UUID: IMSI
- 计费系统: OCS/CGRateS
- 计费规则: play_topup_hotspot

3. 备注

如有疑问, 请随时联系我们。

- HotspotTopup** 服务 - 无线网卡
◦ payment_intent_id

- service_uuid
- imsi
- 帳號
- topup_amount
- 帳號/月/日
- 帳號 - 密碼
 - 帳號“帳號 - 7 ”
 - 帳號topup_amount
 - 帳號 service_id 及 customer_id
- 帳號 - 密碼
 - 帳號
 - 帳號
 - 帳號Stripe payment_intent_id
- 帳號 - 密碼
 - 帳號“ [帳號]”
 - 帳號topup_amount - 帳號
 - 帳號

4. 帳號

帳號 playbook play_topup_hotspot

- 帳號 days
- 帳號 OCS/CGRateS API
- 帳號
- 帳號

帳號 playbook

```
- name: Top up hotspot service
hosts: localhost
tasks:
  - name: Add balance to OCS
    uri:
      url: "{{ ocs_api_url }}/add_balance"
      method: POST
      body:
        imsi: "{{ imsi }}"
        days: "{{ days }}"
        service_uuid: "{{ service_uuid }}"
```

--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

SMS ☐☐☐

```

OCS  Action_Balance_Low  Action_Balance_Out  Action_Balance_Expired

```

□ □ □ □ □ □ □

□ OCS/CGRateS □□□□□□□□□□□□□□□□

□ □ □ □ □ □

- `Action_Balance_Low` - 0000000000000000 2 00
- `Action_Balance_Out` - 0000
- `Action_Balance_Expired` - 0000

□ □

HotspotTopup

SELECT

hotspot_topup_id,
service_uuid,
days,
topup_amount,
status,
payment_intent_id,
created

FROM hotspot_topup

WHERE status = 'Failed'

ORDER BY created DESC;

GET /crm/provision/provision_id/<provision_id>

Stripe

Stripe <<https://dashboard.stripe.com>>

- Stripe PCI 1
- OmniCRM
-
-

-
- IP

- IMSI 認證服務
- 認證服務

認證

- 認證服務 - 認證服務
- 認證服務 - IP 及 IMSI
- 認證服務
- 認證服務 CRM 認證服務

認證

認證

1. 認證 - 認證服務
2. 認證 - 認證服務
3. 認證 **playbook** - 認證 playbook 認證
4. 認證 **OCS** - 認證 OCS API 認證
5. 認證 - 認證 `REACT_APP_TOPUP_PRICE_PER_DAY`

認證

1. 認證 **URL** - 認證
2. 認證 - SMS 認證
3. 認證 - 認證服務
4. 認證 - 認證服務

認證

1. 認證 **Stripe webhooks** - 認證 webhook 認證
2. 認證 - 認證 payment_intent_id
3. 認證 - 認證服務
4. 認證 - 認證服務
5. 認證 - 認證 <5 認證

□□□□

- `payments_process` - □□□□□
- `concepts_provisioning` - □□□□□
- `Payment System Guide <payment_system_guide>` - □□□□□□□□□□
- `payments_transaction` - □□□□
- `payments_invoices` - □□□□

OmniCRM 部署ガイド

OmniCRM 部署ガイド
Ansible による自動化

前提条件

OmniCRM 環境構築

1. OS - CentOS 7
2. ネットワーク - インターネット接続
3. 権限 - root 権限/sudo 権限
4. 言語 - Python 2.7

インストール

OmniCRM 環境構築

インストール

環境構築

- 環境構築
- ネットワーク
- 権限
- 言語

インストール

環境構築

インストール

環境構築

- `product_name` - 상품명
- `product_slug` - API 호출용 URL
- `category` - "standalone" 독립형/통합형
- `service_type` - 서비스 타입
- `enabled` - true/false
- `icon` - UI용 FontAwesome 아이콘
- `comment` - 비고

3. 가격

```
{
  "retail_cost": 15.00,
  "wholesale_cost": 5.00,
  "retail_setup_cost": 0.00,
  "wholesale_setup_cost": 1.00,
  "contract_days": 30
}
```

가격 설명

- 소매가격 £15.00
- 도매가격 £5.00
- 소매설치비용 £10.00 (200% 마진 67%)
- 도매설치비용 -£1.00
- 계약기간 30일

4. 유효성

```
{
  "residential": true,
  "business": false,
  "customer_can_purchase": true,
  "available_from": "2025-01-01T00:00:00Z",
  "available_until": null
}
```

- 유효성 설명

`inventory_items_list` Python `<administration_inventory>`

```
# SIM
inventory_items_list: ["SIM Card", "Mobile Number"]

#
inventory_items_list: ["Modem Router", "Static IP Address"]

#
inventory_items_list: []

# CPE
inventory_items_list: ["Fixed Wireless CPE", "IPv4 Address",
"IPv6 Prefix"]
```

1.

`inventory_items_list`

2.

3.

- -
- - `item_state` "New" "In Stock" "Assigned" "Damaged"
- - `service_id` `customer_id` NULL
- - /

“SIM ”

- ID
- itemtext1 - SIM ICCID
- item_location

4.

- “”
-
-

5. Ansible

“” ID Ansible

```
#  
# - SIM inventory_id: 5001  
# - inventory_id: 5002  
  
# Ansible  
{  
  "product_id": 42,  
  "customer_id": 123,  
  "SIM Card": 5001,          # ID  
  "Mobile Number": 5002,    # ID  
  "access_token": "eyJ..."  
}
```

hostvars[inventory_hostname]['SIM Card'] ID

6.

Ansible ID

```

- name: SIM
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ hostvars[inventory_hostname]['SIM
  Card'] }}"
  method: GET
  headers:
    Authorization: "Bearer {{ access_token }}"
  register: api_response_sim

- name: ICCID IMSI
  set_fact:
    iccid: "{{ api_response_sim.json.itemtext1 }}"
    imsi: "{{ api_response_sim.json.itemtext2 }}"

```

SIM ICCID IMSI HSS

7. “”

```

- name: SIM
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ hostvars[inventory_hostname]['SIM
  Card'] }}"
  method: PATCH
  body:
    {
      "service_id": "{{
  service_creation_response.json.service_id }}",
      "customer_id": "{{ customer_id }}",
      "item_state": "Assigned"
    }

```

- “”
-
-

--	--	--	--	--	--	--	--	--

□□□“□□”□□□□□□□□□□

1. 取得 ID 欄位 `service_id` 值
2. 取得欄位值 OCS 欄位值
3. 取得欄位值

□□□□□□□□

□ □

```
rescue:
  - name: 00000000
    uri:
      url: "{{ crm_config.crm.base_url
}}/crm/inventory/inventory_id/{{ hostvars[inventory_hostname]['SIM
Card'] }}"
      method: PATCH
      body:
        {
          "service_id": null,
          "customer_id": null,
          "item_state": "In Stock"
        }
    when: service_id is defined # 00000000
```

□□□□□□□□□□□□□□□□□□“□□□”□□□

□ □ □ □ □ □ □ □

```
inventory_items_list: "[]"
```

- 000000 - 00000000VPN 00
- 00000000 - 00000000000000
- 000000 - 0000000000

“5GB” inventory_items_list: "[]"

Inventory

```
inventory_items_list: ["SIM Card", "Mobile Number"]
```

- SIM Card ICCID/IMSI eSIM
- MSISDN

Inventory

```
inventory_items_list: ["Modem Router", "Static IP Address"]
```

- MAC CPE
- IP IPv4

Inventory

```
inventory_items_list: ["Fixed Wireless CPE", "IPv4 Address", "IPv6 Prefix"]
```

- CPE
- IPv4 IP
- IPv6 /56 /64

Inventory

VoIP

```
inventory_items_list: ["DID Number"]
```

- DID

Inventory

GPON


```
inventory_items_list: ["'ONT Device', 'GPON Port', 'IPv4 Address',  
'Fiber Drop Cable'"]
```

- ONT 設備名稱
- GPON 設備名稱 OLT 設備名稱
- IPv4 設備名稱 IP
- 設備名稱設備名稱設備名稱設備名稱

設備名稱

```
inventory_items_list: ["'Rental Modem'"]
```

- 設備名稱設備名稱設備名稱
- 設備名稱設備名稱

設備名稱設備名稱

1. 設備名稱

設備名稱設備名稱設備名稱

- 設備 SIM 設備名稱
- 設備 IP 設備名稱
- 設備名稱設備名稱設備名稱

設備名稱設備名稱設備名稱設備名稱

2. 設備名稱

設備名稱設備名稱設備名稱

- 設備 SIM 設備名稱
- 設備名稱
- 設備名稱設備名稱設備名稱
- 設備名稱設備名稱設備名稱設備名稱

3. 設備名稱

□□□□□□

- □ SIM □□□□□□□□
- □□□□□□□□
- □□ CPE □□□□□□□□□□
- □□ IP □□□□□□

4. □□□□

□□□□□□□□□□□□

- □□□□ SIM □□□□
- □□□□□□
- □□□□□□□□□□
- □□□ COGS□□□□□□

5. □□□□

□□□□□□□□□□□□

- □□□□□□SIM □□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□□□□□□□□□

□□□ “□□□□□□”□□□

□□□□

- □□□□□□□□□□□□□□□
- □□□□□“□□”□□□□□□
- □□□□□“□□”□“□□”
- □□□□□□□□□□□□□□□□□□

□□□□□□

1. □□□□□□□□□□□□□□ → □□ → □□

2. 確認する
3. 確認する → 確認 → 確認
4. 確認する“確認”“確認”
5. 確認するservice_id 確認 NULL

確認 確認

確認

- inventory_items_list 確認“[]”
- inventory_items_list 確認 NULL 確認
- 確認“確認”確認

確認

- 確認inventory_items_list: “['Type1', 'Type2']”
- 確認
- 確認 API 確認inventory_items_list

確認 確認“確認”

確認

- 確認
- 確認 ID 確認
- 確認

確認

- 確認hostvars[inventory_hostname]['SIM Card']
- 確認{{ hostvars[inventory_hostname]['SIM Card'] | int }}
- 確認

確認<administration_inventory>

確認 **7**確認

確認

```
{
  "features_list": "20GB 記憶體空間可擴展至30 GB",
  "terms": "30 天內可隨時取消訂閱，恕不退費"
}
```

關鍵字

品牌 - 產品

品牌名稱

- 品牌 - 產品
 - “20GB”“50GB” - 品牌
 - 品牌
- 品牌 - 產品
 - “品牌”
 - “30”
 - 品牌
- 品牌 - 產品
 - 品牌“品牌”→品牌???
 - 品牌
 - 品牌
- 品牌 - 產品
 - 品牌
 - 品牌
 - 品牌
- SEO** 品牌 - 產品
 - 品牌“品牌”→品牌
 - 品牌

品牌 - 產品

品牌

- 品牌 - 產品
 - “30 天” - 31 天
 - “品牌” - 品牌

- วัตถุประสงค์ของการตลาด

2. การตลาด - การตลาด

- “การตลาด” - การตลาด
- “การตลาด”การตลาด - การตลาด
- การตลาด

3. การตลาด - การตลาด

- การตลาดการตลาด
- การตลาดการตลาด
- การตลาด GDPR/การตลาด

4. การตลาด - การตลาด

- “การตลาด” - การตลาด
- “การตลาด” - การตลาด
- “การตลาด” - การตลาด

5. การตลาด - การตลาด

- การตลาดการตลาด
- การตลาดการตลาด
- การตลาดการตลาด

การตลาด

การตลาด“การตลาด”การตลาด 10,000 การตลาด

- การตลาด
- การตลาด“การตลาด.....”
- การตลาด“การตลาด”
- การตลาดการตลาด

การตลาด“การตลาดการตลาดการตลาด”

- การตลาดการตลาด
- การตลาดการตลาด
- การตลาดการตลาด
- การตลาดการตลาด

การตลาด

UI 100Mbps

features_list

1

UI

-
-
- UI
- JSON

2 JSON

"['20GB', ' ', ' ']"

UI JSON JSON JSON

- 100Mbps
- API
-

Python

- UI
- JSON API/

UI

UI

1. 如何部署应用

如何部署应用到 Kubernetes 集群

2. 如何部署应用

如何部署应用到 Kubernetes 集群

- 如何部署应用
- 如何部署应用
- 如何部署应用
- 如何部署应用
- 如何部署应用

3. 如何部署

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

4. 如何部署应用

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

如何部署应用到 Kubernetes 集群

1. 如何部署应用

如何部署应用到 Kubernetes 集群

2. 如何部署

如何部署应用到 Kubernetes 集群

[00] 0000000000

3. 00

0000000000000000000000

00000

- 000 0000000000 50 0000000000
- 000 00000000000000000000
- 000 0000000000000000

00 8000 **Ansible** 0000

```
{
  "provisioning_play": "play_local_mobile_sim",
  "provisioning_json_vars": "{
    \"days\": 30,
    \"data_gb\": 20,
    \"voice_minutes\": \"unlimited\",
    \"sms_count\": \"unlimited\"
  }"
}
```

- provisioning_play - Ansible 00000000 .yaml 0000
- provisioning_json_vars - 0000000000
- 00000000 0mniCRM-API/Provisioners/plays/play_local_mobile_sim.yaml

000000


```
{
  "product_name": "中国联通 20GB",
  "product_slug": "prepaid-mobile-20gb",
  "category": "standalone",
  "service_type": "mobile",
  "enabled": true,
  "icon": "fa-solid fa-sim-card",
  "comment": "中国联通 SIM卡 20GB 全国通用流量",

  "retail_cost": 15.00,
  "wholesale_cost": 5.00,
  "retail_setup_cost": 0.00,
  "wholesale_setup_cost": 1.00,
  "contract_days": 30,

  "residential": true,
  "business": false,
  "customer_can_purchase": true,
  "available_from": "2025-01-01T00:00:00Z",
  "available_until": null,

  "auto_renew": "prompt",
  "allow_auto_renew": true,

  "inventory_items_list": "['SIM Card', 'Mobile Number']",

  "features_list": "[
    '20GB 全国',
    '全国通用流量',
    '全国通用流量',
    '全国',
    '30 天'
  ]",
  "terms": "联通 30 天全国通用流量套餐",

  "provisioning_play": "play_local_mobile_sim",
  "provisioning_json_vars": "{
    \"days\": 30,
    \"data_gb\": 20,
    \"voice_minutes\": \"unlimited\",
    \"sms_count\": \"unlimited\"
  }
```

--	--	--	--	--	--	--	--

1GB

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```

{
  "product_name": "5GB 套餐",
  "product_slug": "5gb-data-boost",
  "category": "addon",
  "service_type": "mobile",
  "enabled": true,
  "icon": "fa-solid fa-plus",
  "comment": "套餐内包含 5GB 流量",

  "retail_cost": 5.00,
  "wholesale_cost": 1.50,
  "retail_setup_cost": 0.00,
  "wholesale_setup_cost": 0.00,
  "contract_days": 0,

  "residential": true,
  "business": true,
  "customer_can_purchase": true,

  "auto_renew": "false",
  "allow_auto_renew": false,

  "inventory_items_list": "[]",
  "relies_on_list": "",

  "features_list": "5GB 流量 7 天",
  "terms": "套餐 7 天内有效",

  "provisioning_play": "play_topup_charge_then_action",
  "provisioning_json_vars": "{
    \"data_gb\": 5,
    \"days\": 7
  }"
}

```

套餐 2 天内有效

套餐 7 天内有效

```

{
  "product_name": "WiFi 6 모뎀 렌탈",
  "product_slug": "wifi6-modem-rental",
  "category": "addon",
  "service_type": "internet",
  "enabled": true,
  "icon": "fa-solid fa-router",
  "comment": "모뎀 렌탈 WiFi 6 모뎀 - 렌탈",

  "retail_cost": 10.00,
  "wholesale_cost": 3.00,
  "retail_setup_cost": 0.00,
  "wholesale_setup_cost": 45.00,
  "contract_days": 30,

  "residential": true,
  "business": true,
  "customer_can_purchase": true,

  "auto_renew": "true",
  "allow_auto_renew": true,

  "inventory_items_list": "['Rental Modem']",
  "relies_on_list": "",

  "features_list": "WiFi 6 (802.11ax) 2.4GHz + 5GHz 40 MHz",
  "terms": "모뎀 렌탈 £150 모뎀 렌탈",

  "provisioning_play": "play_addon_assign_modem",
  "provisioning_json_vars": "{
    \"device_type\": \"modem_router\",
    \"requires_configuration\": true
  }"
}

```

모뎀 렌탈

- `category: "addon"` - 모뎀 렌탈
- `contract_days: 0` 모뎀 렌탈 `30` 모뎀 렌탈 - 모뎀
- `inventory_items_list: "[]"` 모뎀 렌탈 `['Rental Modem']` 모뎀 렌탈 - 모뎀

- 0000000 14 000000

3. 000000

- 000000 item_state = "In Stock"00000000
- 0 item_state = "Damaged"00000000
- 0000000000000000

4. 00000

- 14 000000 £150 000000
- 000000 item_state = "Lost"
- 00000000£450+ 00000

0000000000

0000000000000000000000

- 00000000000000000000
- 000000000000000000000000
- 000000000000 contract_days 0000000

00 20000000

000000“000000 20GB”000000OmniCRM 00 Ansible 000000

0000000

00000 → 00000 → 00000000 ↓ ↓ 00000 ← 00000 ← 00 Ansible 00 ↓ ↓ 00000000 → OCS 00
00 → 00000 → 00000

00000000

1. 0000000

00000000

- 00000“00000”
- 000000000000“000000 20GB”
- 00000000000000

2. 準備

準備するもの

- **SIM** - 仮想SIMカード
◦ 番号“SIM-00123 - ICCID: 8944...”
- 電話番号 - 仮想電話番号
◦ 番号“+44 7700 900123”

準備するもの

3. 準備

準備するもの

- 料金£0.00
- 料金£15.00
- 料金£15.00
- 30 日

準備するもの

☐ 30 日

4. 準備

準備するものAPI

- 番号“”status=1 Provision
- + +
- Ansible
- provision_id UI

5. 準備

準備するもの

準備するもの

```
{
  "days": 30,
  "data_gb": 20,
  "voice_minutes": "unlimited",
  "sms_count": "unlimited"
}
```

□□□□□

```
{
  "product_id": 42,
  "customer_id": 123,
  "SIM Card": 5001,
  "Mobile Number": 5002
}
```

□□□□□

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "initiating_user": 7
}
```

□□□□□ **Ansible** □□□□

```
{
  "product_id": 42,
  "customer_id": 123,
  "SIM Card": 5001,
  "Mobile Number": 5002,
  "days": 30,
  "data_gb": 20,
  "voice_minutes": "unlimited",
  "sms_count": "unlimited",
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "initiating_user": 7
}
```


6. Ansible 실행

play_local_mobile_sim.yaml 실행

Ansible 실행

실행 명령어

SIM 실행

OmniCRM-API/Provisioners/plays/play_local_mobile_sim.yaml

실행

```
- name: SIM 실행
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name:
      block:
        # 1.
        # 2. API
        # 3. API
        # 4. API
        # 5. OCS CGRateS
        # 6. OCS
        # 7. CRM
        # 8.
        # 9.
        # 10.

  rescue:
    #
    # - OCS
    # -
    # -
```

ansible.cfg

1 ansible.cfg

```
- name: ansible builtin include_vars
  ansible.builtin.include_vars:
    file: "../../crm_config.yaml"
    name: crm_config
```

crm_config.yaml

- OCS/CGRateS URL
- CRM URL
-

2 ansible.cfg

```
- name: CRM API
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/product/product_id/{{ product_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
    return_content: yes
  register: api_response_product
```

crm_config.yaml

- GET /crm/product/product_id/42
-
- api_response_product

provisioning_json_vars

-
-
-

- 設定項目を定義

第3項目

```
- name: 設定項目
  set_fact:
    package_name: "{{ api_response_product.json.product_name }}"
    monthly_cost: "{{ api_response_product.json.retail_cost }}"
    setup_cost: "{{ api_response_product.json.retail_setup_cost
  }}"
```

設定項目を定義

第4項目

```
- name: CRM API から SIM 情報を取得
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ hostvars[inventory_hostname]['SIM
  Card'] }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: api_response_sim

- name: SIM 情報の IMSI
  set_fact:
    imsi: "{{ api_response_sim.json.itemtext2 }}"
    iccid: "{{ api_response_sim.json.itemtext1 }}"
```

実行結果

- SIM 情報の ID 5001
- SIM 情報の
 - itemtext1 = ICCIDのSIM 情報
 - itemtext2 = IMSIのSIM 情報
- 実行結果をファイルに保存

実行結果

- IMSI 和 HSS 数据库
- ICCID 数据库
- 数据库 MSISDN 数据库

第 5 步 UUID

```
- name: 生成 UUID 脚本
  set_fact:
    uuid: "{{ 99999999 | random | to_uuid }}"

- name: 生成 UUID
  set_fact:
    service_uuid: "Local_Mobile_SIM_{{ uuid[0:8] }}"
```

数据库

- 生成 UUID
- 生成 Local_Mobile_SIM_a3f2c1d8 和 service_uuid

数据库

- 生成 UUID 和 OCS/CGRateS 数据库
- 数据库
- 数据库

第 6 步 OCS 数据库

```
- name: 0CS
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  headers:
    Content-Type: "application/json"
  body:
    {
      "method": "ApierV2.SetAccount",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "ActionPlanIds": [],
        "ExtraOptions": {
          "AllowNegative": false,
          "Disabled": false
        },
        "ReloadScheduler": true
      }]
    }
register: ocs_create_response
```

0CS

- CGRateS JSON-RPC API
- service_uuid
-
-

0CS

- OCS
-
-

7

```
- name: 数据盘 20GB 数据盘
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV1.AddBalance",
      "params": [{
        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "BalanceType": "*data",
        "Balance": {
          "ID": "DATA_20GB_Monthly",
          "Value": 21474836480,
          "ExpiryTime": "+720h",
          "Weight": 10,
          "DestinationIDs": "*any"
        }
      }]
    }
}
```

数据盘

- 数据盘 20GB 数据盘
- 21474836480 字节 $20 * 1024 * 1024 * 1024$
- 720 小时 30 天
- 10 权重

8 个数据盘

```
- name: 000000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Account": "{{ service_uuid }}",
          "BalanceType": "*voice",
          "Balance": {
            "ID": "VOICE_Unlimited",
            "Value": 999999999,
            "ExpiryTime": "+720h"
          }
        }]
      }
}
```

- 999,999,999 0000000000000000
- 30 0000

9 CRM 00000000

```

- name: 创建 API 接口
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/service/"
    method: PUT
    body_format: json
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      {
        "customer_id": "{{ customer_id }}",
        "product_id": "{{ product_id }}",
        "service_name": "Mobile - {{ phone_number }}",
        "service_type": "mobile",
        "service_uuid": "{{ service_uuid }}",
        "service_status": "Active",
        "service_provisioned_date": "{{ provision_datetime }}",
        "retail_cost": "{{ monthly_cost }}",
        "wholesale_cost": "{{
api_response_product.json.wholesale_cost }}",
        "icon": "fa-solid fa-sim-card"
      }
  register: service_creation_response

```

注意事项

- 创建成功后返回
- 返回 `service_uuid` 用于 OCS
- 创建成功后返回
- 返回结果“成功”
- 返回 `service_id` 用于后续操作

每 10 秒执行一次


```

- name: SIM
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ hostvars[inventory_hostname]['SIM
  Card'] }}"
  method: PATCH
  body_format: json
  headers:
    Authorization: "Bearer {{ access_token }}"
  body:
    {
      "service_id": "{{
service_creation_response.json.service_id }}",
      "customer_id": "{{ customer_id }}",
      "item_state": "Assigned"
    }

```

- SIM
- `service_id` SIM
- “”“”
-

- SIM
-
-

11

```

- name: 0000000000
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/transaction/"
    method: PUT
    body_format: json
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      {
        "customer_id": "{{ customer_id }}",
        "service_id": "{{
service_creation_response.json.service_id }}",
        "title": "{{ package_name }} - 00",
        "description": "0000",
        "retail_cost": "{{ setup_cost }}",
        "wholesale_cost": "{{
api_response_product.json.wholesale_setup_cost }}"
      }

```

00000000

- 00 £0.00 0000000000000000
- 00 £1.00 000000
- 0000000000000000

00 **12**0000000000000000

```

rescue:
  - name: 删除OCS 数据
    uri:
      url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
      method: POST
      body:
        {
          "method": "ApierV2.RemoveAccount",
          "params": [{
            "Account": "{{ service_uuid }}"
          }]
        }

  - name: 删除失败
    fail:
      msg: "删除OCS 失败"

```

部署步骤

- 部署环境准备
- 部署OCS 数据
- 部署“删除”
- 部署删除失败

部署

- OCS 部署
- 部署数据
- 部署删除

部署失败

部署失败

1. OCS 部署CGRateS

- ID Local_Mobile_SIM_a3f2c1d8
- 部署
 - 20GB 数据 30 分钟

- 999M 30
- 999M 30

2. CRM

- ID1234
- John Doecustomer_id: 123
- 20GBproduct_id: 42
- “Mobile - +44 7700 900123”
- UUIDLocal_Mobile_SIM_a3f2c1d8
-
- £15.00£5.00
- £10.00/

3.

- SIM 5001 1234 123
- 5002 1234 123

4.

-
-

5.

-
- 20GB
-
-
-

3

Object Model

Customer has 20GB of storage, 18GB of RAM, and a 5GB of storage

1. Addon

- "Mobile - +44 7700 900123" number
- 20GB of storage, 18GB of RAM, 90% of storage
- "Mobile - +44 7700 900123" number

2. Addon

Customer has 20GB of storage, 18GB of RAM, and a 5GB of storage

- `category = "addon"`
- `service_type = "mobile"` number
- `residential = true` number
- `enabled = true`

Customer has 5GB of storage - £5.00

3. Addon

- "5GB of storage"
- £5.00
- number

4. Addon

Customer has 5GB of storage - £5.00

```
{
  "product_id": 43, # 5GB of storage
  "customer_id": 123,
  "service_id": 1234, # number
  "access_token": "eyJ...",
  "data_gb": 5, # provisioned_json_vars
  "days": 7 # provisioned_json_vars
}
```

□□□□□□□□□□

- □□ `service_id` □□□□□□□□
- □□□□□
- □□□□□□□□□□□□

□□□□□□□□□□□□

□□ **1**□□□□□□□□

```
- name: □ CRM API □□□□□□
  uri:
    url: "http://localhost:5000/crm/service/service_id/{{
service_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: api_response_service

- name: □□□□□□
  set_fact:
    service_uuid: "{{ api_response_service.json.service_uuid }}"
    customer_id: "{{ api_response_service.json.customer_id }}"
```

□□□□

- □□ `service_uuid` □□□□□ OCS □□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□

□□ **2**□□□□□□

```

- name: 创建支付方式
  uri:
    url: "http://localhost:5000/api/payments/methods/default?
customer_id={{ customer_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: api_response_payment_method

- name: 设置支付方式 ID
  set_fact:
    payment_method_id: "{{
api_response_payment_method.json.data.payment_method_id }}"

- name: 创建订单
  uri:
    url: "http://localhost:5000/api/payments/charge"
    method: POST
    body_format: json
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      {
        "customer_id": "{{ customer_id | int }}",
        "amount": 5.00,
        "currency": "USD",
        "payment_method_id": "{{ payment_method_id }}",
        "metadata": {
          "description": "5GB 存储空间",
          "service_id": "{{ service_id | int }}",
          "product_id": "{{ product_id | int }}",
          "invoice": true
        }
      }
  register: charge_response

- name: 断言
  assert:
    that:
      - charge_response.json.success == true

```

测试通过

- Stripe Stripe
- £5.00
- £1.50 Stripe
- Stripe
- Stripe
- Stripe

Stripe

- Stripe
- Stripe
- Stripe

3 OCS

```
- name: Stripe 5GB
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.AddBalance",
        "params": [{
          "Account": "{{ service_uuid }}",
          "BalanceType": "*data",
          "Balance": {
            "ID": "DATA_5GB_Boost_{{ uuid }}",
            "Value": 5368709120,
            "ExpiryTime": "+168h",
            "Weight": 20
          }
        }]
      }
```

Stripe

- Stripe 5GB 5368709120
- 168 7

- 20% discount - 20% discount

20% discount

- 2GB 25 % discount
- 5GB 7 % discount
- 7GB
- 20% discount

4% discount

```
- name: 20% discount
  uri:
    url: "http://localhost:5000/crm/transaction/"
    method: PUT
    body_format: json
    headers:
      Authorization: "Bearer {{ access_token }}"
    body:
      {
        "customer_id": "{{ customer_id }}",
        "service_id": "{{ service_id }}",
        "title": "5GB 20%",
        "description": "20% 5GB 20% 7 %",
        "retail_cost": 5.00,
        "wholesale_cost": 1.50
      }
}
```

20% discount

- £5.00 20% discount
- £1.50 20% discount
- 20% discount

20% discount

1. 20% discount
2. 20% discount
3. 20% discount OCS 20%

4. 在 CRM 中创建

5. 创建产品时指定 7GB

配置

- 创建产品时 £15 每月 £5 押金
- 创建产品时 £5 押金 £1.50 每月

创建产品时

创建产品时指定 7GB

配置

```
{
  "product_name": "7GB 10GB 产品",
  "category": "addon",
  "retail_cost": 10.00,
  "contract_days": 30,
  "auto_renew": "true",
  "provisioning_play": "play_topup_charge_then_action"
}
```

创建产品时

```
- name: []
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body:
    {
      "method": "ApierV1.SetActionPlan",
      "params": [{
        "Id": "ServiceID_{{ service_uuid }}__ProductID_{{
product_id }}__MonthlyRenewal",
        "ActionPlan": [{
          "ActionsId": "Action_{{ product_slug }}",
          "Years": "*any",
          "Months": "*any",
          "MonthDays": "*any",
          "WeekDays": "*any",
          "Time": "00:00:00",
          "Weight": 10
        }],
        "Overwrite": false,
        "ReloadScheduler": true
      }]
    }
}
```

--	--	--	--	--	--

- OCS 製品ページ
- 30 日間 `Action_{{ product_slug }}`
- 製品ページから製品ページへ
- 製品ページから

□□□□□

- [illegible]

第 4 章

本章主要介绍...

本章学习目标

通过本章学习，应达到以下目标：

1. 了解 - 掌握“...”
2. 了解 - 掌握...
3. 了解 - 掌握...
4. 了解 - 掌握...

本章重点难点

1. 重点难点

- ...
- “...”
- “...”
- ...

2. 重点难点

本章重点难点如下：

- “...”
- 7-30 ...
- ...
- ...

3. 重点难点

本章重点难点如下：

```
{
  "action": "deprovision",
  "service_id": 1234,
  "customer_id": 123,
  "service_uuid": "Local_Mobile_SIM_a3f2c1d8"
}
```

[[service.deprovisioning_play]]

4. Ansible

```

- name: 测试OCS接口
  hosts: localhost
  tasks:
    - name: 测试OCS接口
      uri:
        url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
        method: POST
        body:
          {
            "method": "ApierV2.SetAccount",
            "params": [{
              "Account": "{{ service_uuid }}",
              "ExtraOptions": { "Disabled": true }
            }]
          }

    - name: 测试CRM接口
      uri:
        url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
        method: POST
        body:
          {
            "method": "ApierV1.RemoveActionPlan",
            "params": [{
              "Id": "ServiceID_{{ service_uuid }}__*"
            }]
          }

    - name: 测试CRM接口
      uri:
        url: "http://localhost:5000/crm/service/{{ service_id }}"
        method: PATCH
        body:
          {
            "service_status": "Deactivated",
            "service_deactivate_date": "{{ current_datetime }}"
          }

    - name: 测试CRM接口
      uri:
        url: "http://localhost:5000/crm/inventory/inventory_id/{{
sim_card_id }}"
        method: PATCH

```

```
body:
{
  "service_id": null,
  "customer_id": null,
  "item_state": "Decommissioned"
}
```

□□□□□□

1. □□ **OCS** □□ - □□□□□□□□□□
2. □□□□□□ - □□□□□□
3. □□ **CRM** □□ - □□□“□□□”□□□□□
4. □□□□ - SIM □□□“□□□”□□□□□□□□□□

5. □□□□□□

□□□□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□ CRM □□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□
- OCS □□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□

- □□□□□□□□□□□□
- □□ OCS □□□□□□□□
- □□□□□
- □□□□□□□□□□□□

```
- name: 0000
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body:
      {
        "method": "ApierV2.SetAccount",
        "params": [{
          "Account": "{{ service_uuid }}",
          "ExtraOptions": { "Disabled": true }
        }]
      }
}
```

0000000000000000

- 00000000
- 0000 OCS 00
- 0000
- 00000000

```
- name: 00 OCS 00
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body:
      {
        "method": "ApierV2.RemoveAccount",
        "params": [{
          "Account": "{{ service_uuid }}"
        }]
      }
}
```

0000000000000000

000000000000

000000

- `enabled: true` - 有効
- `enabled: false` - 無効

設定

- 有効 `enabled: false` の場合
- 有効期限
- 有効期限/有効範囲
- 有効範囲

状態

状態

- `New` - 新規
- `In Stock` - 在庫
- `Assigned` - 割り当て
- `Decommissioned` - 廃止
- `Damaged` - 破損

状態

状態

- SIM 番号
- 有効期限30 日間
- 有効範囲

設定

設定

- 有効
- 有効期限
- 有効範囲
- 有効範囲

Mailjet と OmniCRM の統合

OmniCRM は **Mailjet** を統合して、マーケティング自動化を実現します。

概要

Mailjet の設定

- 設定項目 - 必要な項目を確認
- 認証 - Mailjet の API キーを設定
- 送信頻度 - 10 分以内で Mailjet に送信
- 送信先 - 送信先を設定
- 送信内容 - 送信内容を設定

設定

Mailjet の `OmniCRM-API/crm_config.yaml` に `mailjet` を追加

設定例

```
mailjet:  
  api_key: your_mailjet_api_key  
  api_secret: your_mailjet_api_secret
```

API の取得

1. <https://www.mailjet.com> にアクセス
2. 登録 → **API** を取得
3. **API** キーとシークレットを取得
4. `crm_config.yaml` に設定

メール送信

OmniCRM の **10** 項目のメール送信の項目

- **from_email** - 送信元メールアドレス
- **from_name** - 送信元名前
- **template_id** - Mailjet の ID
- **subject** - 件名

メール送信

メール送信

メール送信

```
api_crmCommunicationCustomerWelcome:  
  from_email: "support@yourcompany.com"  
  from_name: "Your Company Support"  
  template_id: 5977509  
  subject: "メール送信"
```

メール送信

- メール送信
- メール送信
- メール送信

メール送信

- {{ var:customer_name }} - メール送信
- {{ var:email }} - メール送信
- {{ var:company_name }} - メール送信
- {{ var:login_url }} - メール送信
- {{ var:support_url }} - メール送信

メール送信

顧客からの請求書

```
api_crmCommunicationCustomerInvoice:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 6759851
  subject: "請求書 - "
```

請求書

- 請求書の送信
- 請求書の確認
- 請求書の支払い

請求書の送信

- {{ var:customer_name }} - 顧客名
- {{ var:invoice_number }} - 請求書 ID/番号
- {{ var:invoice_date }} - 請求書の日付
- {{ var:due_date }} - 支払期限
- {{ var:total_amount }} - 請求書の総額
- {{ var:invoice_url }} - 請求書の PDF へのリンク
- {{ var:pay_url }} - 支払いのリンク

請求書の確認

請求書の PDF をダウンロード

請求書の支払い

請求書の支払い方法

```
api_crmCommunicationCustomerInvoiceReminder:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 6759852
  subject: "請求書 - 支払いのお願い"
```

メール送信

- 送信先 X 送信元
- 送信件名
- 送信本文

メール送信パラメータ

- {{ var:customer_name }}
- {{ var:invoice_number }}
- {{ var:due_date }}
- {{ var:days_overdue }}
- {{ var:total_amount }}
- {{ var:pay_url }}

メール送信結果

メール送信結果の取得

```
api_crmCommunicationUserWelcome:  
  from_email: "admin@yourcompany.com"  
  from_name: "Your Company Admin"  
  template_id: 5977510  
  subject: "メール送信"
```

メール送信

- 送信先
- 送信元

メール送信

- {{ var:user_name }} - 送信元
- {{ var:email }} - 送信先
- {{ var:role }} - 送信元
- {{ var:login_url }} - 送信元
- {{ var:temp_password }} - 送信元

- `{{ var:support_email }}` - IT 部門

メール本文

メール本文のテンプレート

```
api_crmCommunicationUserPasswordReset:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977511
  subject: "パスワードリセット"
```

メール本文

- パスワードリセットリンク
- パスワードリセットの期限

メール本文

- `{{ var:user_name }}`
- `{{ var:reset_url }}` - パスワードリセットリンク 1 回
- `{{ var:expiry_time }}` - パスワードリセットの期限

メール本文

パスワードリセットリンク 1 回

パスワードリセット

パスワードリセットの期限

```
api_crmCommunicationUserPasswordResetSuccess:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977512
  subject: "パスワードリセット完了"
```

メール本文

- 密碼重置通知
- 密碼重置通知

密碼重置

- {{ var:user_name }}
- {{ var:change_date }} - 密碼重置/日期
- {{ var:ip_address }} - 密碼 IP 地址
- {{ var:support_email }} - 密碼重置通知

密碼重置

密碼重置通知

```
api_crmCommunicationUserPasswordChange:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977513
  subject: "密碼重置"
```

密碼重置

- 密碼重置/密碼重置
- 密碼重置

密碼重置

- {{ var:user_name }}
- {{ var:change_date }}
- {{ var:changed_by }} - “密碼”密碼重置
- {{ var:support_email }}

密碼重置

密碼重置通知

```
api_crmCommunicationEmailVerification:
  from_email: "noreply@yourcompany.com"
  from_name: "XXXX"
  template_id: 5977514
  subject: "XXXXXXXXXXXX"
```

XXXX

- XXXXXXXXXXXXXXX
- XXXXXXXXXXXXXXX
- XXXXXXXXXXXXXXX

XXXXXXXX

- {{ var:user_name }}
- {{ var:verification_url }} - XXXXXXXXXXXXXXX
- {{ var:verification_code }} - XXXXXXXXXXXXXXX

XXXXXX

XXXXXXXXXX/XXXXXXXXXX

```
api_crmCommunicationsBalanceExpired:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977515
  subject: "XXXXXXXXXXXX"
```

XXXX

- XXXXXXX
- XXXXXXXXXXXXXXX
- XXXXXXX

XXXXXXXX

- {{ var:customer_name }}
- {{ var:service_name }} - XXXXXXX

- {{ var:expiry_date }}
- {{ var:balance_type }} - “”“”“”“”
- {{ var:renewal_url }} - /

```
api_crmCommunicationsBalanceLow:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977516
  subject: ""
```

- 20%
- OCS
-

- {{ var:customer_name }}
- {{ var:service_name }}
- {{ var:current_balance }}
- {{ var:threshold }}
- {{ var:balance_type }}
- {{ var:topup_url }} -

Mailjet

Mailjet

1 Mailjet

1. Mailjet

2. 200 OK → 200 OK
3. 200 OK
4. 200 OK

2 200 OK

Mailjet 200 OK HTML 200 OK

200 OK

- 200 - 200 OK
- 200 - 200 OK {{ var:customer_name }} {{ var:user_name }} 200 OK
- 200 - 200 OK
- 200 - 200 OK
- 200 - 200 OK/200 OK
- 200 - 200 OK

200 OK

```

<!DOCTYPE html>
<html>
<head>
  <style>
    body { font-family: Arial, sans-serif; }
    .button { background-color: #4CAF50; color: white;
padding: 14px 28px; }
  </style>
</head>
<body>
  ![Logo](https://yourcompany.com/logo.png)

  <h2>○○○○○○</h2>

  <p>○○ {{ var:user_name }}</p>

  <p>○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○</p>

  <a href="{{ var:reset_url }}" class="button">○○○○</a>

  <p>○○○○ {{ var:expiry_time }} ○○○</p>

  <p>○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○</p>

  <hr>
  <p style="font-size: 12px; color: #666;">
    ○○○ | support@yourcompany.com<br/>
    123 ○○○○○○○○○
  </p>
</body>
</html>

```

3 ○○○○○ ID

1. Mailjet ○○○○
2. ○○ ID○○○○○○ 5977509
3. ○○ ID ○○○ crm_config.yaml

4. 環境構築

1. Mailjet アカウントを作成する
2. API キーを取得する
3. Mailjet の設定を確認する
4. 環境変数を設定する

5. OmniCRM のインストール

設定ファイル `crm_config.yaml`

```
mailjet:
  api_key: your_api_key
  api_secret: your_secret

api_crmCommunicationUserPasswordReset:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977511
  subject: "パスワードリセット"
```

OmniCRM API を再起動する

```
cd OmniCRM-API
sudo systemctl restart omnicrm-api
```

確認

OmniCRM のインストールが完了したら、Mailjet の

確認

- 確認
- 確認
- 確認
- 確認

- 設定

設定

設定項目

- 設定
- 設定/設定
- 設定/設定

設定

設定

- 設定 API Mailjet 設定 ID
- 設定
 - `crm_config.yaml` 設定 `api_key` `api_secret`
 - Mailjet 設定
 - ID Mailjet
 - API 設定

設定

- 設定 OmniCRM 設定
- 設定
 - 設定
 - `{{ var:variable_name }}` 設定
 - OmniCRM API 設定
 - Mailjet 設定

PDF 設定

- PDF 設定
- 設定
 - 設定
 - 設定

- PDF 15MB Mailjet
- PDF

Mailjet

- API
- - Mailjet API 200
 - OmniCRM-API
 -
 -

- `administration_configuration` - Mailjet
- `payments_invoices` -
- `authentication_flows` -
- `customer_care` -
- - Mailjet API

- Mailjet <<https://dev.mailjet.com/>>
- Mailjet API <<https://dev.mailjet.com/email/reference/>>



OmniCRM 50

OmniCRM API

OmniCRM API を Prometheus で監視するための Prometheus 監視ジョブ定義を `/crm/metrics` エンドポイントから Prometheus に提供します。

監視

監視対象

- `crm` - CRM システムのメトリクス
- `stripe` - Stripe のメトリクス
- `ocs` - OCS/CGRateS のメトリクス
- `mailjet` - Mailjet API のメトリクス
- **HTTP** エンドポイント - API の可用性と応答時間

設定

URL: `http://your-omnicrm-api:5000/crm/metrics`

監視: Prometheus 監視

監視: Prometheus 監視ジョブ定義を Prometheus に提供します。

インストール

1. Ansible

Ansible を使用して Prometheus 監視ジョブ定義を Ansible Playbooks に Provisioning System に Ansible Playbooks

omnicrm_provision_jobs_total

監視: Prometheus

{}:

- `status` - {}: `success`, `failed`, `running`

{}: {}

omnicrm_provision_job_duration_seconds

{}: {}

{}:

- `playbook` - {} Ansible {}
- `status` - {}: `success`, `failed`

{}: [1, 5, 10, 30, 60, 120, 180, 300, 600] {}1{}10{}

{}: {}

omnicrm_provision_jobs_active

{}: {}

{}: {}

omnicrm_provision_tasks_total

{}: {}

{}:

- `playbook` - Ansible {}
- `status` - {}: `ok`, `failed`, `ignored`

{}: {} Ansible {}

omnicrm_provision_errors_total

{}: {}

{}:

- `error_type` - 可能: `fatal`, `task_failed`, `timeout`
- `playbook` - Ansible 名前

例: 実行結果を出力する

2. 実行結果

実行結果は、OmniCRM の SQLAlchemy 実行結果を System Architecture

omnicrm_db_query_duration_seconds

例: 実行

例:

- `operation` - SQL 操作: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- `table` - テーブル名

例: `[0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0]`

例: SQLAlchemy 実行結果

omnicrm_db_pool_size

例: 実行

例: 実行結果

omnicrm_db_pool_overflow

例: 実行

例: 実行結果

omnicrm_db_pool_connections_checked_out

例: 実行

例: 実行結果

omnicrm_db_errors_total

単位: 個

項目:

- `error_type` - エラータイプ
- `operation` - 実行操作: `connection_error`, `query_error`, ...

単位: 個

単位: 個

3. OCS/CGRateS

OCS/CGRateS API に関する情報は [Billing Overview](#) と [CGRateS Integration](#) を参照してください。

omnicrm_ocs_api_calls_total

単位: 個

項目:

- `method` - OCS API 名: `GetBalance`, `SetBalance`, `SetAccount`, `RemoveAccount`, ...
- `status` - ステータス: `success`, `failed`

単位: OCS/CGRateS API 呼び出し回数

omnicrm_ocs_api_duration_seconds

単位: 秒

項目:

- `method` - OCS API 名: `GetBalance`, `SetBalance`, `GetAccount`, ...

単位: [0.1, 0.25, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0] 秒

📄: OCS/CGRateS API 📄📄📄📄📄📄 API 📄📄📄📄📄📄📄📄

omnicrm_ocs_api_errors_total

📄: 📄📄

📄:

- `method` - 📄📄 OCS API 📄📄
- `error_type` - 📄📄📄: `timeout`, `connection_error`, `json_error`, `http_error`, 📄📄

📄: OCS/CGRateS API 📄📄📄📄 OCS API 📄📄📄📄📄📄📄📄

omnicrm_ocs_balance_queries_total

📄: 📄📄📄📄

📄:

- `query_type` - 📄📄📄📄: `single_account`, `multiple_accounts`

📄: 📄 OCS 📄📄📄📄📄📄📄📄📄📄📄📄📄📄

omnicrm_ocs_action_plan_operations_total

📄: 📄📄

📄:

- `operation` - 📄📄📄📄: `create`, `remove`, `query`

📄: 📄📄📄📄📄📄📄 CGRateS 📄📄📄📄📄📄📄📄📄📄📄📄

4. Stripe 📄📄📄

Stripe 📄📄📄📄📄📄📄📄📄📄📄📄📄📄 [Payment System Guide](#) 📄 [Payment Methods](#)📄

omnicrm_stripe_api_calls_total

型: 型

型:

- operation - Stripe 型: create_customer, charge, refund, update_payment_method, 型
- status - 型: success, failed

型: Stripe API 型

omnicrm_stripe_api_duration_seconds

型: 型

型:

- operation - Stripe 型

型: [0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0] 型

型: Stripe API 型

omnicrm_stripe_api_errors_total

型: 型

型:

- operation - 型 Stripe 型
- error_type - 型: card_declined, network_error, api_error, 型

型: Stripe API 型

omnicrm_stripe_payment_amount_cents

型: 型

型:

- payment_type - 型: charge, refund

例: Stripe 決済失敗の総数

omnicrm_stripe_payment_failures_total

例: 100

例:

- reason - 理由: card_declined, insufficient_funds, expired_card, ...

例: Stripe 決済失敗の総数

5. Mailjet 連携

Mailjet 連携に関する情報は [Mailjet Integration](#) を参照してください。

omnicrm_mailjet_api_calls_total

例: 100

例:

- email_type - メールタイプ: welcome, user_welcome, invoice, notification
- status - ステータス: success, failed

例: Mailjet API 呼び出し @track_mailjet_call の総数

omnicrm_mailjet_api_duration_seconds

例: 100

例:

- email_type - メールタイプ

例: [0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0]

例: Mailjet API 呼び出しエラーの総数 Mailjet API 呼び出しエラーの総数

omnicrm_mailjet_api_errors_total

例: 例

例:

- email_type - 例
- error_type - 例

例: Mailjet API 例

omnicrm_mailjet_emails_sent_total

例: 例

例:

- email_type - 例
- template_id - Mailjet 例 ID

例: 例 Mailjet 例 API 例

omnicrm_mailjet_email_recipients_total

例: 例

例:

- email_type - 例

例: 例

6. 例

例 Provisioning System

omnicrm_background_jobs_total

例: 例

例:

- `job_type` - 可能値: `playbook_single`, `playbook_chain`, `periodic_task`

例: 背景ジョブタイマーの背景ジョブ

omnicrm_background_jobs_active

例: 1

例:

- `job_type` - 可能値

例: 背景ジョブタイマーの背景ジョブ

omnicrm_background_job_duration_seconds

例: 100

例:

- `job_type` - 可能値
- `status` - 可能値: `success`, `failed`

例: [1, 5, 10, 30, 60, 120, 180, 300, 600, 1800, 3600] 1分1秒1分

例: 背景ジョブタイマーの背景ジョブ

omnicrm_background_job_errors_total

例: 100

例:

- `job_type` - 可能値
- `error_type` - 可能値

例: 背景ジョブタイマーの背景ジョブ

7. Flask HTTP 메트릭들

이 섹션에서는 `prometheus-flask-exporter`가 제공하는 API 및 HTTP 메트릭

`flask_http_request_duration_seconds`

타입: 가우시안

라벨:

- `method` - HTTP 메소드: `GET`, `POST`, `PUT`, `DELETE`, ...
- `endpoint` - Flask 엔드포인트
- `status` - HTTP 상태

단위: 각 API 엔드포인트 HTTP 요청당 소요 시간

`flask_http_request_total`

타입: 카운터

라벨:

- `method` - HTTP 메소드
- `endpoint` - Flask 엔드포인트
- `status` - HTTP 상태

단위: 모든 API 엔드포인트 HTTP 요청 수

`flask_http_request_exceptions_total`

타입: 카운터

라벨:

- `method` - HTTP 메소드
- `endpoint` - Flask 엔드포인트

단위: HTTP 요청당 예외 발생 횟수

8. API 錯誤訊息

OmniCRM API 錯誤訊息

omnicrm_api_errors_total

類型: 數字

欄位:

- `endpoint` - API 端點
- `error_type` - 錯誤類型
- `status_code` - HTTP 狀態碼

單位: 錯誤次數

omnicrm_api_auth_failures_total

類型: 數字

欄位:

- `auth_method` - 認證方法: `jwt`, `api_key`, `ip_whitelist`
- `reason` - 失敗原因

單位: 失敗次數

9. 系統資訊

omnicrm_api

類型: 物件

欄位:

- `version` - API 版本
- `environment` - 環境: `production`, `staging`, `development`

說明: OmniCRM API 系統資訊

□□□□

update_db_pool_metrics(engine)

□ 30 □□□□□□□□□□□□□□□□□□

Prometheus □□

□□□□

□ OmniCRM □□□□ `prometheus.yml` □

```
scrape_configs:
  - job_name: 'omnicrm-api'
    scrape_interval: 15s
    scrape_timeout: 10s
    static_configs:
      - targets: ['omnicrm-api:5000']
    metrics_path: '/crm/metrics'
```

□□□□

□□□□□□

```
- alert: HighProvisionFailureRate
  expr: |
    (
      rate(omnicrm_provision_jobs_total{status="failed"}[5m]) /
      rate(omnicrm_provision_jobs_total[5m])
    ) > 0.1
  for: 5m
  labels:
    severity: warning
  annotations:
    summary: "High Provision Failure Rate"
    description: "{{ $value | humanizePercentage }} High Provision Failure Rate"
```

High Provision Failure Rate

```
- alert: DatabasePoolExhausted
  expr: omnicrm_db_pool_overflow > 0
  for: 2m
  labels:
    severity: critical
  annotations:
    summary: "Database Pool Exhausted"
    description: "Database Pool Exhausted"
```

Database Pool Exhausted

```
- alert: SlowDatabaseQueries
  expr: |
    histogram_quantile(0.99,
      rate(omnicrm_db_query_duration_seconds_bucket[5m])
    ) > 1.0
  for: 5m
  labels:
    severity: warning
  annotations:
    summary: "Slow Database Queries"
    description: "99th Percentile of Database Query Duration is {{ $value }}s"
```

OCS API Alerts

```
- alert: OCSAPIDown
  expr: |
    (
      rate(omnicrm_ocs_api_calls_total{status="failed"}[5m]) /
      rate(omnicrm_ocs_api_calls_total[5m])
    ) > 0.5
  for: 2m
  labels:
    severity: critical
  annotations:
    summary: "OCS API 异常"
    description: "{{ $value | humanizePercentage }} % OCS API 异常"
```

Stripe 异常

```
- alert: StripePaymentFailures
  expr: rate(omnicrm_stripe_payment_failures_total[5m]) > 5
  for: 5m
  labels:
    severity: warning
  annotations:
    summary: "Stripe 异常"
    description: "{{ $value }} 异常"
```

异常

异常

1. 异常 - 异常

- 异常 > 10%
- 异常
- OCS/CGRateS API 异常
- Stripe 异常

2. 監控指標 - 監控 KPI

- Stripe 監控
- 監控
- 監控

3. 監控 - 監控

- API 監控
- 監控
- API 監控

4. 監控 - 監控

- 監控
- 監控
- API 監控

Prometheus 監控:

- 15 - 監控15 監控
- 90 - 監控5 監控
- 1 - 監控1 監控

Thanos Cortex VictoriaMetrics 監控

metrics 監控 nginx 監控

:

```
curl http://localhost:5000/crm/metrics
```

📄 **Prometheus** 📄📄📄:

```
# 📄 Prometheus 📄📄  
http://prometheus:9090/targets
```

📄📄📄📄

📄📄📄📄📄📄📄📄

- 📄📄📄📄📄📄📄📄
- Stripe 📄📄📄📄📄📄
- OCS 📄📄📄📄📄📄📄📄

📄📄📄📄

📄 Prometheus 📄📄📄📄📄📄📄📄📄📄📄📄📄📄

```
# 📄📄📄📄📄📄📄  
count by ( __name__ ) ( { __name__ =~ ".+" } )
```

📄 >10,000 📄📄📄📄📄📄📄📄📄📄

📄📄📄

📄📄: 34📄31 📄📄📄 + 3 📄📄📄📄 Flask 📄📄

📄📄:

- 📄📄: 20
- 📄📄: 5
- 📄📄: 8

- 容器: 1

容器:

- 容器: 5 容器
 - 容器: 5 容器
 - OCS/CGRateS: 5 容器
 - Stripe: 5 容器
 - Mailjet: 5 容器
 - 容器: 4 容器
 - HTTP/Flask: 3 容器
 - 容器: 1 容器
 - 容器: 2 容器
-

容器

- [System Architecture](#) - 容器
- [Provisioning System](#) - 容器
- [Ansible Playbooks](#) - 容器
- [API Documentation](#) - API 容器
- [Billing Overview](#) - 容器
- [Payment System Guide](#) - Stripe 容器
- [Mailjet Integration](#) - 容器
- [Administration Configuration](#) - 容器

Payment System Guide

OmniCRM 提供了一個完整的支付系統指南，包括 Stripe 和 PayPal 的集成指南。此指南旨在幫助您了解如何配置和集成 OmniCRM 的支付系統。

Payment System Guide <payment_system_guide> Billing Overview <billing_overview> Payment Processing <payments_process> Invoices <payments_invoices>

Table of Contents

Table of Contents

- Introduction - Overview of Stripe and PayPal integration with OmniCRM
- Getting Started - Stripe and PayPal Setup
- Configuration - System Settings
- Integration - Stripe and PayPal Integration
- Self-Care Portal - Self-Care Portal <self_care_portal> Integration
- Invoices - Invoices and Billing Cycle

Table of Contents

- Stripe and PayPal
 - Stripe
 - Stripe Integration
 - Stripe Integration
 - Stripe Integration
 - PayPal
 - PayPal Integration

Table of Contents OmniCRM

Table of Contents OmniCRM

- Stripe and PayPal
 - Stripe
 - Stripe Integration
 - Stripe Integration
 - Stripe Integration
 - PayPal
 - PayPal Integration

- 顧客情報
- 顧客ID/名前
- 顧客住所/郵便番号
- 顧客電話番号/メールアドレス

顧客情報

顧客ID

1. 顧客 **Customers** → [顧客ID]
2. 顧客 **Billing** 情報
3. 顧客 **Payment Methods** 情報

顧客住所

顧客電話番号

顧客メールアドレス

顧客登録日から60日間有効

顧客ID

顧客IDは一意の識別子

信用卡

- 信用卡 - 信用卡卡號“XXXX”“XXXX Visa”
- 信用卡 - 信用卡卡號
- 信用卡 - 信用卡/到期MM/YYYY 日期
- 信用卡 - 信用卡卡號
- 信用卡 - 信用卡卡號

信用卡

信用卡卡號 : 信用卡卡號

- 信用卡 - 信用卡卡號
- 信用卡 - 信用卡卡號

信用卡

信用卡 "Add Payment Method" 信用卡卡號

信用卡

信用卡卡號 Stripe Elements 信用卡卡號 PayPal SDK 信用卡卡號

信用卡

- 銀行 - カード番号CVC Stripe 入力
- カード番号 - カード番号
- 月/年 - カード有効期限

銀行

- カード - カード番号

銀行

- カード番号入力欄に iframe Stripe Elements / PayPal SDK
- OmniCRM カード番号入力欄
- PCI DSS カード番号入力欄
- カード番号入力欄

銀行

銀行 "Add Payment Method" 入力

1. カード番号
 - カード番号入力欄
 - カード番号入力欄
 - カード CVC 入力
2. カード
 - カード番号入力欄 OmniCRM
 - カード番号入力欄 pm_1A2B3C4D Stripe
 - カード番号 OmniCRM
3. カード番号
 - OmniCRM カード番号入力欄
 - カード番号入力欄
 - カード番号入力欄 OmniCRM

銀行

銀行

銀行 Visa カード 1234

0000000000000000

00000000

- 00000000000000000000
- 00000000000000000000
- 0000000000000000

0000000000

0000000000

- 00000000
- 0000
- 00000
- 0000000000000000

00000000

1. 00000000000000000000
2. 00000000 000000 : 0
3. 00 **"Set as Default"**
4. 00000000

Visa 00 5678 00000000000000

0000000000000000

000000

00000000

00000000000000000000

刪除付款方法

請按照以下步驟操作：

步驟一：登入帳戶

1. 前往 [登入頁面](#)
2. 輸入您的帳號及密碼
3. 點擊 **"Delete"**

步驟二：選擇要刪除的付款方法

在您的帳戶頁面，

您將看到所有已儲存的付款方法。

例如：Visa 1234 12/2026

⚠️ 請確保您選擇的是您想要刪除的付款方法。

[圖示] [刪除按鈕]

點擊 **"Delete Payment Method"** 按鈕。

步驟三：確認刪除

系統將顯示

確認對話框，請確認您是否要刪除該付款方法。

點擊

- 確認按鈕以完成刪除。
- 取消按鈕以返回上一步。
- 幫助 - 如果您遇到任何問題，請點擊此處。

XXXXXXXXXX

OmniCRM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX

XXX **Billing → Expiring Cards** XXXXXXXXXXXXX

XX XX XX XXXXXX John Smith Visa ****1234 02/2025 12** X XX **Acme Corp**
MC5678 03/2025 45 X XX **Jane Doe Amex**9012 01/2025** XXX XX

💡💡XXX

- XXXX - XXXX 30/60/90 XXXXX
- XXXX - XXXXX
- XXXX - XXXXXXXXXXXXX

XXX

- XX - XXXXXXXXXXXXXXXXXXXXX
- XX - XXXXXXXXXXXXXXXXXXXXX Mailjet

Mailjet

- 100 **60** 1 - 100000
- 100 **30** 1 - 100000
- 100 **7** 1 - 10000
- 100 - 10000000

□ □

□ □ □ □ □ □ □ □

Mailjet ☐

- 
- 
- 
- 

```
integrations_mailjet
```

--	--	--	--	--	--	--	--	--

□□□□□□

1. **Account Settings**
2. **Payment Methods**
3. **Billing → Payment Methods**
4. **"Add Payment Method"**
5. **Enter Card Details**
6. **Save Method**
7. **Confirmation/Success**

□ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □

1. 目標を定める
2. 目標 **Billing → Payment Methods**
3. 目標達成のための具体的な行動計画を立てる
4. 目標達成のためのリソースを確保する
5. 目標達成のためのモチベーションを維持する
6. 目標達成のためのフィードバックを受ける

目標

目標達成のための具体的な行動計画を立てる

- 目標達成のための具体的な行動計画を立てる
- 目標達成のためのリソースを確保する
- 目標達成のためのモチベーションを維持する

目標達成のためのリソースを確保する

目標達成のためのモチベーションを維持する

目標達成のためのフィードバックを受ける

1. 目標を定める
 - 目標達成のための具体的な行動計画を立てる
 - 目標達成のためのリソースを確保する
 - 目標達成のためのモチベーションを維持する
 - 目標達成のためのフィードバックを受ける
2. 目標 **Billing**
 - 目標達成のための具体的な行動計画を立てる
 - 目標達成のためのリソースを確保する
 - 目標達成のためのモチベーションを維持する
 - 目標達成のためのフィードバックを受ける
3. 目標達成のための具体的な行動計画を立てる
 - 目標達成のためのリソースを確保する
 - 目標達成のためのモチベーションを維持する
 - 目標達成のためのフィードバックを受ける

目標達成のためのフィードバックを受ける

□ 1-3 □□□□□□

- □□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□

□ 4-7 □□

- □□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□

□ 8-14 □□

- □□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□

□ 15 □□□□□

- □□□□□□□□□□□□
- □□□SIM □□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□□□□

- □□□□ **30** □ □□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□

1. □□□□□□□□□□□□

2. 商家
3. 商家類型
4. 商家地址
5. 商家電話

商家類型

商家

OmniCRM 商家類型

1. 商家 → 商家類型
2. 商家類型 → 商家
3. 商家 **OmniCRM** → 商家類型
4. 商家 → 商家地址

OmniCRM 商家

```
{
  "vendor": "stripe",
  "vendor_payment_method_id": "pm_1A2B3C4D5E6F",
  "payment_type": "card",
  "brand": "visa",
  "last4": "1234",
  "exp_month": 12,
  "exp_year": 2026,
  "name": "John Smith",
  "nickname": "Personal Card",
  "is_default": true
}
```

OmniCRM 商家

- 商家
- CVV/CVC
- 商家
- PIN

PCI 要求

要遵守 PCI 要求，请确保：

- 所有 **PCI** 数据 - 存储在 OmniCRM 中
- 传输数据 - 使用安全的 iframe 或
- 数据 - 存储在安全的数据库中
- 数据 - 使用 HTTPS/TLS

参考 [Payment System Guide <payment_system_guide>](#) 以了解有关 PCI 要求的详细信息。

实施步骤

步骤 1：评估当前状态

评估当前状态：

1. 数据存储在何处
2. 数据如何传输
3. 数据是否加密
4. 是否使用 Stripe 支付网关
5. 是否使用安全的数据库
6. 是否使用安全的 API
7. 是否使用安全的网络
8. 是否使用安全的设备

步骤 2：实施安全措施

实施安全措施：

1. 确保所有数据在传输和存储时都加密
2. 确保所有数据在传输和存储时都加密
3. 参考 **Billing → Payment Methods**
4. 确保所有数据在传输和存储时都加密
5. 参考 **"Add Payment Method"**

6. 日期 12/2028 日期
7. 日期日期
8. 日期
9. 日期日期

日期 3 日期日期日期日期

日期 日期日期“日期日期”

1. 日期日期
2. 日期日期日期日期
3. 日期日期日期日期 01/2025
4. 日期“日期日期日期日期”
5. 日期日期日期日期
6. 日期 **Customers → [Customer] → Billing**
7. 日期 **"Add Payment Method"**
8. 日期日期日期日期
9. 日期日期
10. 日期日期
11. 日期日期
12. 日期日期“日期日期”

日期 4 日期日期日期日期

日期 日期日期日期日期日期

1. 日期日期日期 Visa 日期 1111
2. 日期日期日期日期
3. 日期日期 Mastercard 日期 2222 日期日期
4. 日期日期 Amex 日期 3333 日期日期
5. 日期日期日期日期日期 Mastercard
6. 日期 Visa 日期日期日期

📅 5📅📅📅📅📅📅📅📅📅📅

📅📅 📅📅📅📅

1. 📅📅📅📅 **Billing → Expiring Cards**
2. 📅📅📅“📅📅📅 30 📅”
3. 📅📅 15 📅📅📅📅📅📅📅📅
4. 📅📅📅 → **"Send Reminder Emails"**
5. Mailjet 📅📅📅📅📅📅📅📅📅
6. 📅📅📅📅📅📅📅📅
7. 📅📅📅📅📅📅📅
8. 📅📅📅📅📅📅📅
9. 📅📅📅📅📅📅📅

📅📅📅📅

“📅📅📅📅📅📅📅”

- 📅📅 📅📅📅📅📅📅📅📅📅📅📅📅📅📅📅📅
- 📅📅📅
 - 📅📅📅📅📅
 - 📅📅📅📅📅📅📅
 - 📅📅📅📅📅📅
 - 📅📅📅📅📅📅📅📅📅

“📅📅📅📅📅📅”📅📅📅📅

- 📅📅 📅📅📅 API 📅📅📅📅📅
- 📅📅📅
 - 📅📅📅📅📅
 - 📅📅📅📅📅
 - 📅📅📅📅📅📅📅📅📅📅
 - 📅📅📅📅📅📅📅📅📅
 - 📅📅📅📅📅📅📅📅📅

プラットフォーム構築

- 開発環境構築
- 認証
◦ OAuth2.0
◦ JWT

プラットフォーム構築“コア機能”

- 開発環境構築
- 認証
◦ OAuth2.0
◦ JWT
◦ OAuth2.0

プラットフォーム

- 開発環境構築
- 認証
◦ OAuth2.0
◦ JWT
◦ OAuth2.0

プラットフォーム

- 開発環境構築 SDK API 構築
- 認証
◦ OAuth2.0
◦ JWT
◦ OAuth2.0
◦ OAuth2.0 SDK 構築
◦ OAuth2.0/JWT

プラットフォーム

- 開発 Mailjet 構築
- 認証

- 編輯 `crm_config.yaml` 編輯 Mailjet 設定
- 測試發送電子郵件
- 測試發送簡訊
- 編輯 Mailjet 測試電子郵件

測試

測試

- 測試發送電子郵件
- 測試 2 測試電子郵件
- 測試 30 測試電子郵件
- 測試/測試電子郵件
- 測試電子郵件“`Visa`”“`Amex`”
- 測試電子郵件

測試

- 測試電子郵件
- 測試電子郵件/SMS/測試電子郵件
- 測試電子郵件
- 測試電子郵件
- 測試電子郵件
- 測試電子郵件 £0.01 測試

測試

- 測試電子郵件
- 測試 60/30/7 測試電子郵件
- 測試電子郵件/測試電子郵件
- 測試 Mailjet 測試電子郵件
- 測試電子郵件
- 測試電子郵件

目次

- `Payment System Guide <payment_system_guide>` - API 概要
- `payments_process` - 決済処理
- `payments_invoices` - 決済明細
- `features_topup_recharge` - チャージ機能
- `basics_payment` - 基本決済
- `customer_care` - カスタマーサポート

OmniCRM API

API

OmniCRM API is a RESTful API that allows you to integrate your application with OmniCRM. It provides endpoints for managing users, accounts, and transactions. Supported payment gateways include Stripe, PayPal, Square, Adyen, and Braintree.

API documentation is available at [API Documentation](#).

Ansible Playbook for automating API integration. See [Playbooks](#) for more details.

API

1. [API Overview](#)
2. [API Endpoints](#)
3. [API Authentication](#)
4. [API Rate Limits](#)
5. [API Error Handling](#)
6. [API Security](#)
7. [API Testing](#)
8. [API Examples](#)

API

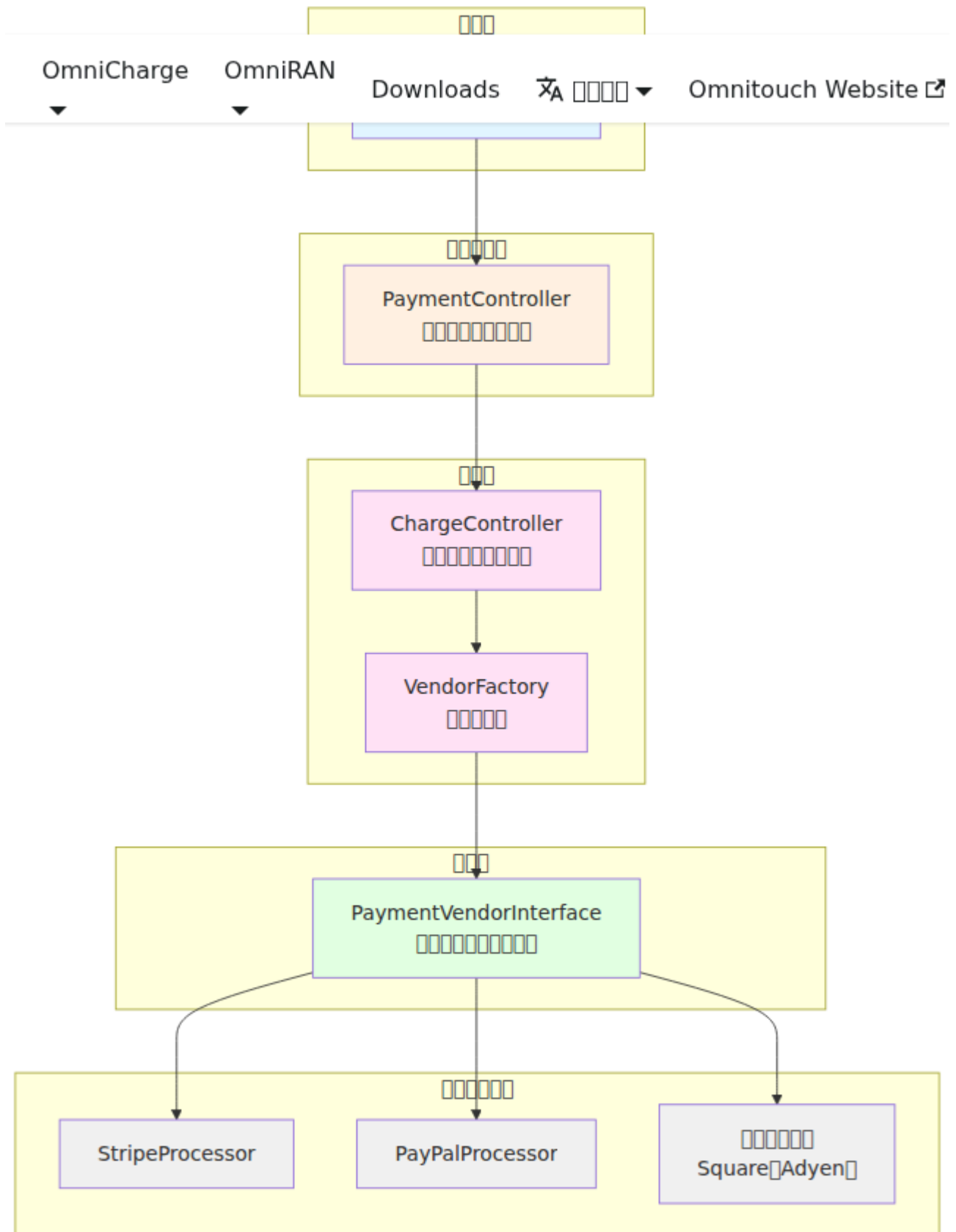
API Endpoints

API endpoints are used to manage users, accounts, and transactions.

- [API Endpoints](#)
- [API Authentication](#)

- 0 0000000000
- 0 0000000000API000000

□□□



Payment Vendor

Payment Vendor: Square, Adyen, Braintree, OmniCRM

Payment Vendor: Stripe, PayPal, **API Key**

Payment

Payment

Payment

Payment

Payment

Payment

API Key

Payment

- PaymentVendorInterface
- VendorFactory
-
- API

Payment

```
# Square
curl -X POST /api/payments/methods \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "customer_id": 123,
    "vendor": "square",
    "payment_token": "sq_xxxxx"
  }'
```

API

1-2



Stripe

Stripeは、オンライン決済を簡単にできる3Dセキュア決済サービス

1. Stripe API

- <https://stripe.com> へ
- 登録 → API
- 公開鍵 pk_live_... / pk_test_...
- 秘密鍵 sk_live_... / sk_test_...

例

- 秘密鍵 pk_test_ / sk_test_
- 公開鍵 pk_live_ / sk_live_
- APIキー

2. 設定

例 `crm_config.yaml`

```
payment_vendors:
  stripe:
    api_key: "sk_live_YOUR_SECRET_KEY_HERE"
    publishable_key: "pk_live_YOUR_PUBLISHABLE_KEY_HERE"
```

例

```
export STRIPE_SECRET_KEY="sk_live_YOUR_SECRET_KEY_HERE"
export STRIPE_PUBLISHABLE_KEY="pk_live_YOUR_PUBLISHABLE_KEY_HERE"
```

3. 設定

例 `.env`

```
REACT_APP_STRIPE_PUBLISHABLE_KEY=pk_live_YOUR_PUBLISHABLE_KEY_HERE
```

□ □

PayPal

PayPalPayPal

11

```
crm_config.yaml
```

```
payment_vendors:
  paypal:
    client_id: "AXx_YOUR_CLIENT_ID_HERE"
    client_secret: "ELx_YOUR_CLIENT_SECRET_HERE"
    mode: "live" # or "sandbox"  
```

PCI

OmniCRM ☐ ☐ ☐ PCI ☐ ☐ ☐

- Stripe Elements PayPal Card Fields
- OmniCRM
-
- PCI

--	--	--	--	--	--

OmniCRM ██████████ Stripe ██████████ API ██████████
 ████

□ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

PaymentAuthorization

PaymentAuthorization

```
{
  "authorization_id": 301,
  "customer_id": 123,
  "payment_method_id": 789,
  "vendor": "stripe", // 供应商
  "vendor_authorization_id": "auth_xxx", // 供应商ID
  "amount": 200.00,
  "currency": "USD",
  "status": "authorized", // 'authorized',
  'captured', 'released'
  "authorized_at": "2025-12-27T10:00:00Z",
  "expires_at": "2026-01-03T10:00:00Z",
  "meta": {}
}
```

PaymentCapture

PaymentCapture

```
{
  "capture_id": 103,
  "authorization_id": 301,
  "customer_id": 123,
  "payment_method_id": 789,
  "vendor": "stripe",
  "vendor_transaction_id": "ch_xxx", // 供应商ID
  "amount": 200.00,
  "currency": "USD",
  "status": "succeeded", // 'succeeded', 'failed',
  'refunded'
  "captured_at": "2025-12-27T10:30:00Z",
  "vendor_response": {}, // 供应商响应
  "meta": {}
}
```

WalletAccount

钱包账户信息

```
{
  "wallet_account_id": 456,
  "customer_id": 123,
  "balance": 150.50,
  "currency": "USD",
  "auto_recharge_enabled": true,
  "auto_recharge_amount": 100.00,
  "auto_recharge_threshold": 10.00,
  "low_balance_warning_threshold": 10.00
}
```

WalletLedger

钱包交易记录

```
{
  "ledger_id": 501,
  "customer_id": 123,
  "wallet_account_id": 456,
  "transaction_type": "credit",           // 'credit', 'debit',
  'refund', 'adjustment'
  "amount": 100.00,
  "balance_before": 150.50,
  "balance_after": 250.50,
  "currency": "USD",
  "description": "Card top-up",
  "reference_type": "payment_capture",   // 支付记录
  "reference_id": 103,
  "meta": {},
  "created_at": "2025-12-27T10:35:00Z"
}
```

钱包

钱包余额**0000**

- 00000000000000000000000000000000
- 00000000000000000000000000000000
- 000000000000 - 0000000000000000

□ □

```
// 1. 成功のレスポンス
{
  "transaction_id": 7001,
  "invoice_id": 12345,
  "retail_cost": 100.00,
  "title": "成功"
}

// 2. 失敗のレスポンス
{
  "transaction_id": 7002,
  "invoice_id": 12345,
  "retail_cost": -100.00,
  "title": "失敗: 12345",
  "payment_type": "stripe_capture",
  "payment_reference": "ch_xxxxx"
}

// 成功$0000 → 失敗$0000
```

```

payment_reference payment_type
payment_time

```

- invoice_id Hayden
- /
-

1111

[illegible]

API

Stripe API を利用して、決済処理を実行する方法を説明します。

準備

- Stripe アカウントを作成する
- Stripe API キーを取得する
- Stripe CLI をインストールする
- Stripe 開発者ツールを使用する
 - Stripe 開発者ツールをインストールする
 - Stripe 開発者ツールを使用する

Stripe API を利用して、決済処理を実行する方法を説明します。Stripe API を利用して、決済処理を実行する方法を説明します。

Stripe API

Stripe API の URL は `/api/payments/` です。

Stripe API の API キーは `**STRIPE_API_KEY**` です。

Stripe API

Stripe API の API キーは `**STRIPE_API_KEY**` です。

Stripe API の API キーは `**STRIPE_API_KEY**` です。

Stripe API の API キーは `**STRIPE_API_KEY**` です。

```
curl -X POST https://your-domain.com/api/payments/methods \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "vendor": "stripe",
  "payment_token": "pm_xxxxx",
  "is_default": false,
  "card_nickname": "My Visa Card"
}'
```

□□□□

```
{
  "customer_id": 123,
  "vendor": "stripe", // 'stripe', 'paypal'□□□□
  "payment_token": "pm_xxxxx", // SDK□□□□□□□□
  "is_default": false, // □□□□□□□□□□
  "card_nickname": "My Visa Card" // □□□□□□□□
}
```

□□□201□□□□□□

```
{
  "success": true,
  "message": "成功",
  "data": {
    "payment_method_id": 789,
    "customer_id": 123,
    "vendor": "stripe",
    "payment_type": "card",
    "card_brand": "visa",
    "card_last4": "4242",
    "card_exp_month": 12,
    "card_exp_year": 2025,
    "card_nickname": "My Visa Card",
    "is_default": false,
    "status": "active"
  }
}
```

成功

成功

GET /api/payments/methods?customer_id={id}

```
curl -X GET "https://your-domain.com/api/payments/methods?
customer_id=123" \
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

```
{
  "success": true,
  "data": [
    {
      "payment_method_id": 789,
      "customer_id": 123,
      "vendor": "stripe",
      "payment_type": "card",
      "card_brand": "visa",
      "card_last4": "4242",
      "is_default": true
    },
    {
      "payment_method_id": 790,
      "customer_id": 123,
      "vendor": "paypal",
      "payment_type": "paypal",
      "paypal_email": "user@example.com",
      "is_default": false
    }
  ]
}
```

□□□□□□□□

□□□□□□□□□□

□□□ GET /api/payments/methods/default?customer_id={id}

□□□

```
curl -X GET "https://your-domain.com/api/payments/methods/default?customer_id=123" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

□□□ 200 OK □□


```
{
  "success": true,
  "data": {
    "payment_method_id": 789,
    "vendor": "stripe",
    "payment_type": "card",
    "card_brand": "visa",
    "card_last4": "4242",
    "is_default": true
  }
}
```

🔍 🔍 🔍 🔍 🔍 🔍 🔍 🔍

🔍 🔍 🔍 🔍 🔍 🔍 🔍 🔍

🔍 🔍 🔍 PUT /api/payments/methods/set-default

🔍 🔍 🔍

```
curl -X PUT https://your-domain.com/api/payments/methods/set-
default \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "customer_id": 123,
    "payment_method_id": 790
  }'
```

🔍 🔍 🔍 200 OK 🔍 🔍

```
{
  "success": true,
  "message": "🔍 🔍 🔍 🔍 🔍 🔍 🔍 🔍",
  "data": {
    "payment_method_id": 790,
    "is_default": true
  }
}
```

DELETE

DELETE

DELETE /api/payments/methods/{payment_method_id}?customer_id={id}

curl

```
curl -X DELETE "https://your-domain.com/api/payments/methods/789?customer_id=123" \
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

```
{
  "success": true,
  "message": "Payment method deleted"
}
```

API

API

1. API

API

POST /api/payments/charge

curl

```
curl -X POST https://your-domain.com/api/payments/charge \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 50.00,
  "currency": "USD",
  "payment_method_id": 789,
  "metadata": {
    "order_id": "12345"
  }
}'
```

□□□□

```
{
  "customer_id": 123,
  "amount": 50.00,
  "currency": "USD", // □□□USD
  "payment_method_id": 789, // □□ - □□□□□□□□□□
  "vendor": "stripe", // □□□□payment_token□□□
  "payment_token": "pm_xxxxx", // □□ - □□□□□
  "save_method": false, // □□□□□□□□□□□□□□
  "metadata": {
    "order_id": "12345"
  }
}
```

□□□200 OK□□

```
{
  "success": true,
  "message": "成功",
  "data": {
    "transaction_id": "ch_xxxxx",
    "capture_id": 101,
    "amount": 50.00,
    "currency": "USD",
    "status": "succeeded"
  }
}
```

2. 準備作業

準備作業のフローチャート

API: `POST /api/payments/invoice`

手順

1. 準備
2. 準備
3. 準備  準備
4. 準備
5. 準備

準備

```
curl -X POST https://your-domain.com/api/payments/invoice \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 200.00,
  "payment_method_id": 789,
  "metadata": {
    "invoice_id": 12345,
    "description": "□□□□"
  }
}'
```

200 OK

```
{
  "success": true,
  "message": "成功",
  "data": {
    "customer_id": 123,
    "service_amount": 200.00,
    "routing_mode": "hybrid",
    "initial_balance": 150.00,
    "wallet_portion_used": 150.00, // wallet 使用金額
    "card_portion_used": 50.00,    // 信用卡使用金額
    "charged_amount": 50.00,
    "wallet_credited": 50.00,
    "wallet_debited": 200.00,
    "final_balance": 0.00,
    "payment_method_used": true
  }
}
```

3. □□□□□□□□

□ □

Playbook Ansible playbook

1

👉 `POST /api/payments/authorize/hold`

👉👉👉👉👉👉

1. 📄📄📄📄
2. 📄📄📄📄 - 📄📄📄
3. 📄📄📄📄📄📄
4. 📄📄📄📄📄📄📄📄📄📄📄

👉👉

```
curl -X POST https://your-domain.com/api/payments/authorize/hold \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 200.00,
  "payment_method_id": 789,
  "use_wallet": true,
  "metadata": {
    "reservation_id": "RES-001"
  }
}'
```

👉👉👉

```
{
  "customer_id": 123,
  "amount": 200.00,
  "currency": "USD",
  "payment_method_id": 789,           // 📄 - 📄📄📄📄📄📄
  "vendor": "stripe",                // 📄📄📄payment_token📄📄
  "payment_token": "pm_xxxxx",      // 📄 - 📄📄📄
  "save_method": false,
  "use_wallet": true,                // 📄📄📄📄📄📄📄📄📄true📄
  "metadata": {
    "reservation_id": "RES-001"
  }
}
```

200 OK

```
{
  "success": true,
  "message": "Payment successful",
  "data": {
    "authorization_id": 301,
    "vendor_authorization_id": "auth_xxxxx",
    "amount": 200.00,
    "currency": "USD",
    "status": "authorized",
    "wallet_balance": 150.00,
    "wallet_to_use": 150.00,           // wallet balance
    "card_amount": 50.00,             // card amount
    "message": "Card $50 and wallet $150 used for $200 payment"
  }
}
```

2. Capture

POST /api/payments/capture/{authorization_id}

Request

1. Authorization ID - 301
2. Amount - 200
3. Currency - USD
4. Card amount / Wallet amount

Response

```
curl -X POST https://your-domain.com/api/payments/capture/301 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "amount": 200.00,
  "metadata": {
    "invoice": true,
    "title": "测试",
    "description": "3天",
    "wholesale_cost": 100.00,
    "contract_days": 3,
    "send_email": true
  }
}'
```

测试

```
{
  "amount": 200.00, // 金额 - 测试金额
  "metadata": {
    "invoice": true, // 是否生成发票
    "create_transaction": true, // 是否创建交易
    "title": "测试",
    "description": "3天",
    "wholesale_cost": 100.00,
    "contract_days": 3,
    "send_email": true // 是否发送邮件
  }
}
```

测试200 OK


```
{
  "success": true,
  "message": "成功",
  "data": {
    "capture_id": 103,
    "transaction_id": "ch_xxxxx",
    "authorization_id": 301,
    "amount": 200.00,
    "currency": "USD",
    "status": "succeeded",
    "wallet_credit": { // 信用
      "ledger_id": 401,
      "amount": 50.00
    },
    "wallet_debit": { // 借方
      "ledger_id": 402,
      "amount": 200.00
    },
    "transaction": { // invoice=true
      "transaction_id": 7001
    },
    "invoice": { // invoice=true
      "invoice_id": 12345,
      "invoice_number": "INV-2025-000001"
    }
  }
}
```

3

POST /api/payments/release/{authorization_id}

```
curl -X POST https://your-domain.com/api/payments/release/301 \
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

```
{
  "success": true,
  "message": "成功",
  "data": {
    "authorization_id": 301,
    "vendor_authorization_id": "auth_xxxxx",
    "status": "released",
    "released_at": "2025-12-27T10:45:00Z"
  }
}
```

成功

4. 決済

決済/決済履歴

POST /api/payments/topup

決済 → 決済履歴 → 決済

決済

```
curl -X POST https://your-domain.com/api/payments/topup \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 30.00,
  "payment_method_id": 789,
  "service_uuid": "svc-uuid-123",
  "imsi": "123456789012345",
  "days": 30,
  "metadata": {
    "is_rental": false
  }
}'
```

決済/決済履歴

```
curl -X POST https://your-domain.com/api/payments/topup \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 1,
  "amount": 5.00,
  "vendor": "stripe",
  "payment_token": "pm_xxxxx",
  "service_uuid": "hotspot-uuid",
  "imsi": "123456789012345",
  "days": 1,
  "metadata": {
    "is_rental": true,
    "billing_email": "user@example.com"
  }
}'
```

200 OK

```
{
  "success": true,
  "message": "Success",
  "data": {
    "transaction_id": "ch_xxxxx",
    "authorization_id": 302,
    "capture_id": 104,
    "amount": 30.00,
    "status": "succeeded",
    "provision_result": {
      "success": true,
      "topup_result": {...},
      "service_uuid": "svc-uuid-123",
      "imsi": "123456789012345",
      "days": 30
    },
    "payment_method_saved": false // false
  }
}
```

5. 請求を送信

このセクションでは、API へのリクエストを送信する方法について説明します。

リクエストは `POST /api/payments/rental` に対して送信されます。

リクエストの例:

```
curl -X POST https://your-domain.com/api/payments/rental \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "beneficiary_customer_id": 456,
    "charge_customer_id": 123,
    "amount": 75.00,
    "payment_method_id": 789,
    "service_description": "レンタル",
    "metadata": {
      "rental_agreement_id": "RA-001"
    }
  }'
```

このリクエストは、API へのリクエストを送信するための例です。

```
curl -X POST https://your-domain.com/api/payments/rental \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "beneficiary_customer_id": 456,
    "amount": 75.00,
    "vendor": "stripe",
    "payment_token": "pm_xxxxx",
    "service_description": "レンタル",
    "metadata": {
      "billing_email": "renter@example.com"
    }
  }'
```

リクエストが成功すると、`200 OK` が返されます。

```
{
  "success": true,
  "message": "成功",
  "data": {
    "transaction_id": "ch_xxxxx",
    "amount": 75.00,
    "payment_method_saved": false // 是否保存
  }
}
```

6. 退款

POST /api/payments/refund

curl

```
curl -X POST https://your-domain.com/api/payments/refund \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "transaction_id": "ch_xxxxx",
  "vendor": "stripe",
  "amount": 50.00,
  "reason": "customer_request"
}'
```

响应

```
{
  "transaction_id": "ch_xxxxx", // 交易ID
  "vendor": "stripe", // 'stripe', 'paypal'
  "amount": 50.00, // 金额 - 退款金额
  "reason": "customer_request" // 原因
}
```

返回 200 OK

```
{
  "success": true,
  "message": " refunded ",
  "data": {
    "refund_id": "re_xxxxx",
    "amount": 50.00,
    "status": "succeeded"
  }
}
```

API

API URL `/api/wallet/`

API

GET `/api/wallet/balance?customer_id={id}`

API

```
curl -X GET "https://your-domain.com/api/wallet/balance?
customer_id=123" \
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

```
{
  "customer_id": 123,
  "balance": 150.50,
  "currency": "USD"
}
```

API

API

GET /api/wallet/info?customer_id={id}

```
curl -X GET "https://your-domain.com/api/wallet/info?customer_id=123" \
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

```
{
  "customer_id": 123,
  "wallet": {
    "wallet_account_id": 456,
    "balance": 150.50,
    "currency": "USD",
    "auto_recharge_enabled": true,
    "auto_recharge_amount": 100.00,
    "auto_recharge_threshold": 10.00,
    "low_balance_warning_threshold": 10.00
  }
}
```

POST /api/wallet/topup

/PayPal →

```
curl -X POST https://your-domain.com/api/wallet/topup \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer YOUR_API_KEY" \  
-d '{  
  "customer_id": 123,  
  "amount": 100.00,  
  "payment_method_id": 789  
}'
```

□□□□

```
{  
  "customer_id": 123,  
  "amount": 100.00,  
  "currency": "USD", // □□□USD  
  "payment_method_id": 789 // □□ - □□□□□□□□  
}
```

□□□200 OK□□

```
{  
  "success": true,  
  "payment": {  
    "transaction_id": "ch_xxxxx",  
    "amount": 100.00  
  },  
  "wallet": {  
    "ledger_id": 503,  
    "balance_after": 250.50  
  },  
  "message": "□□□□100.00□□□□"  
}
```

□□□□□□

□□□□□□□□□□□□


```
GET /api/wallet/transactions?customer_id={id}&limit={n}&offset={n}&type={type}
```

```
curl -X GET "https://your-domain.com/api/wallet/transactions?customer_id=123&limit=50&offset=0&type=credit" \
-H "Authorization: Bearer YOUR_API_KEY"
```

- `customer_id` ID
- `limit` 50
- `offset` 0
- `type` 'credit' 'debit' 'refund' 'adjustment'

200 OK

```
{
  "customer_id": 123,
  "count": 2,
  "transactions": [
    {
      "ledger_id": 501,
      "transaction_type": "credit",
      "amount": 100.00,
      "balance_before": 150.50,
      "balance_after": 250.50,
      "description": "□□□",
      "reference_type": "payment_capture",
      "reference_id": 103,
      "created_at": "2025-12-27T10:35:00Z"
    },
    {
      "ledger_id": 502,
      "transaction_type": "debit",
      "amount": 50.00,
      "balance_before": 250.50,
      "balance_after": 200.50,
      "description": "□□□□",
      "reference_type": "service_charge",
      "reference_id": 789,
      "created_at": "2025-12-27T11:00:00Z"
    }
  ]
}
```

API

POST	/api/payments/methods	
GET	/api/payments/methods?customer_id={id}	
GET	/api/payments/methods/default?customer_id={id}	
PUT	/api/payments/methods/set-default	
DELETE	/api/payments/methods/{id}?customer_id={id}	

POST	/api/payments/charge	
POST	/api/payments/invoice	
POST	/api/payments/topup	
POST	/api/payments/authorize/hold	
POST	/api/payments/capture/{id}	
POST	/api/payments/release/{id}	
POST	/api/payments/rental	
POST	/api/payments/refund	

API

Method	API Path	Description
GET	/api/wallet/balance?customer_id={id}	Get balance
GET	/api/wallet/info?customer_id={id}	Get wallet info + card
POST	/api/wallet/topup	Top up wallet
GET	/api/wallet/transactions?customer_id={id}	Get transactions

PayPal

Method	API Path	Description
POST	/api/payments/paypal/vault/setup-token	Card Fields SDK PayPal vault
POST	/api/payments/paypal/vault/finalize	PayPal vault Finalize
POST	/api/payments/paypal/vault/update-setup-token	3DS/SCA Update setup token

API

1

30

```
# 创建发票并支付
curl -X POST https://your-domain.com/api/payments/invoice \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "customer_id": 123,
    "amount": 99.99,
    "metadata": {
      "service_id": 456,
      "contract_days": 30,
      "description": "30天会员"
    }
  }'
```

返回

```
{
  "success": true,
  "message": "支付成功",
  "data": {
    "service_amount": 99.99,
    "wallet_portion_used": 50.00,           // 钱包$50
    "card_portion_used": 49.99,           // 卡片$49.99
    "final_balance": 0.00
  }
}
```

2. 创建订单

创建订单并支付

```
# 充值接口
curl -X POST https://your-domain.com/api/payments/topup \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "customer_id": 123,
    "amount": 149.99,
    "payment_method_id": 789,
    "service_uuid": "fiber-svc-uuid",
    "imsi": "123456789012345",
    "days": 30,
    "metadata": {
      "service_type": "fiber_internet"
    }
  }'
```

步骤

1. 充值\$149.99
2. 充值成功
3. 充值成功 → 充值
4. 充值成功 → 充值成功

3. 充值成功

充值成功WiFi充值成功

```
# 充值
curl -X POST https://your-domain.com/api/payments/topup \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 1,
  "amount": 5.00,
  "vendor": "stripe",
  "payment_token": "pm_xxxxx",
  "service_uuid": "hotspot-downtown",
  "imsi": "999999999999999",
  "days": 1,
  "metadata": {
    "is_rental": true,
    "billing_email": "user@example.com",
    "hotspot_location": "充值"
  }
}'
```

注意

- 充值金额
- 充值次数
- 充值有效期
- 充值地点

4 +

```
# 1. authorize - authorize
curl -X POST https://your-domain.com/api/payments/authorize/hold \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 500.00,
  "use_wallet": true,
  "metadata": {
    "reservation_id": "RES-2025-001"
  }
}'

# 2. status {"authorization_id": 301, "status": "authorized"}

# 2a. capture - capture
curl -X POST https://your-domain.com/api/payments/capture/301 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "metadata": {
    "invoice": true,
    "title": "Stripe - 301",
    "description": "302 Stripe 12 27 30",
    "send_email": true
  }
}'

# 2b. release - release
curl -X POST https://your-domain.com/api/payments/release/301 \
-H "Authorization: Bearer YOUR_API_KEY"
```

5 Stripe

Visa


```
# 1 Stripe.js Stripe
# const {token} = await stripe.createToken(cardElement);

# 2
curl -X POST https://your-domain.com/api/payments/methods \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "vendor": "stripe",
  "payment_token": "pm_xxxxx",
  "is_default": true,
  "card_nickname": "Visa"
}'
```

6

\$100

```
curl -X POST https://your-domain.com/api/wallet/topup \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 100.00
}'
```

```
{
  "success": true,
  "payment": {
    "transaction_id": "ch_xxxxx",
    "amount": 100.00
  },
  "wallet": {
    "balance_after": 250.50
  },
  "message": "成功100.00元支付"
}
```

成功

成功

成功400元支付

```
{
  "error": "成功50.00元支付200.00"
}
```

成功400元支付

```
{
  "error": "成功"
}
```

成功400元支付

```
{
  "error": "customer_idamount"
}
```

🔍 404 Not Found

```
{
  "error": "🔴 999 🟡"
}
```

🔍 500 Internal Server Error

```
{
  "error": "🔴🔴🔴🔴🔴🔴🔴"
}
```

🔍🔍🔍🔍🔍🔍🔍🔍

🔍🔍🔍

🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍

1. 🔍🔍🔍🔍🔍 **Stripe/PayPal** 🔍

🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍

🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍 PayPal 🔍🔍🔍

🔍🔍 `POST /api/payments/refund`

🔍🔍

```
curl -X POST https://your-domain.com/api/payments/refund \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "transaction_id": "ch_xxxxx",
  "vendor": "stripe",
  "amount": 100.00,
  "reason": "customer_request"
}'
```

111

- Stripe/PayPal
- 5-10/PayPal
-
- **PaymentCapture**

--	--	--	--	--

- 
- 
- 
- 

2. 問題

[illegible][illegible]

111

- 00000000
- 0000
- 0000000
- 0WalletLedger0000000000

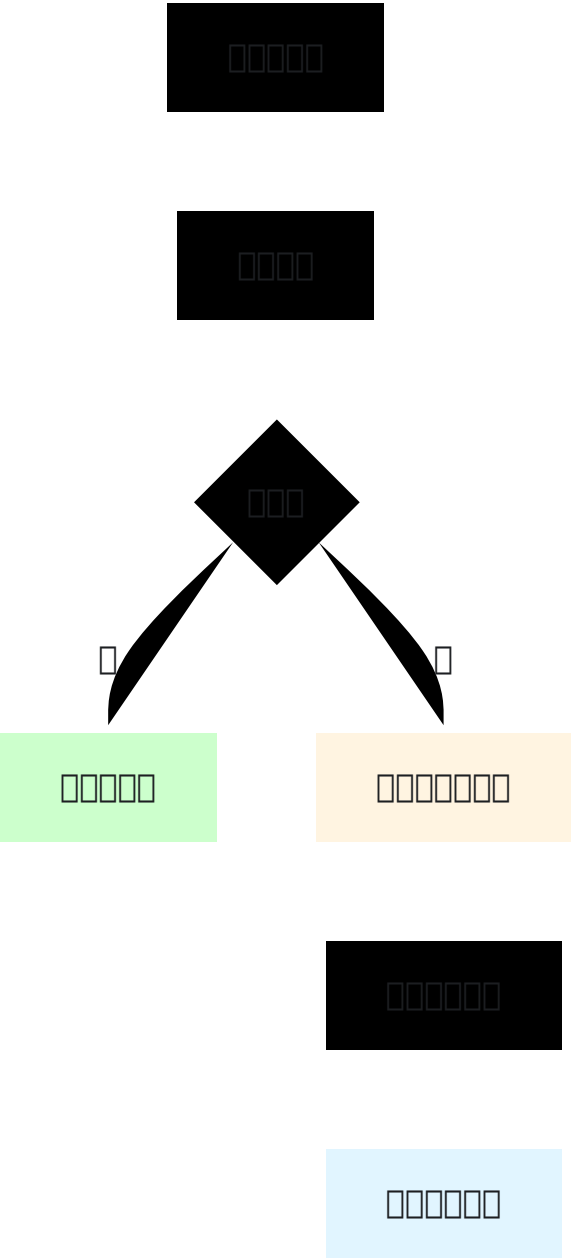
□□□□□

- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1. 購買\$100禮券
2. 購買
3. 購買
4. 購買\$100禮券5-10 + 購買
 - 購買\$100禮券
 - 購買
 - 購買
 - 購買

購買禮券

購買禮券

1\$1 + \$10

購買\$1禮券\$10禮券

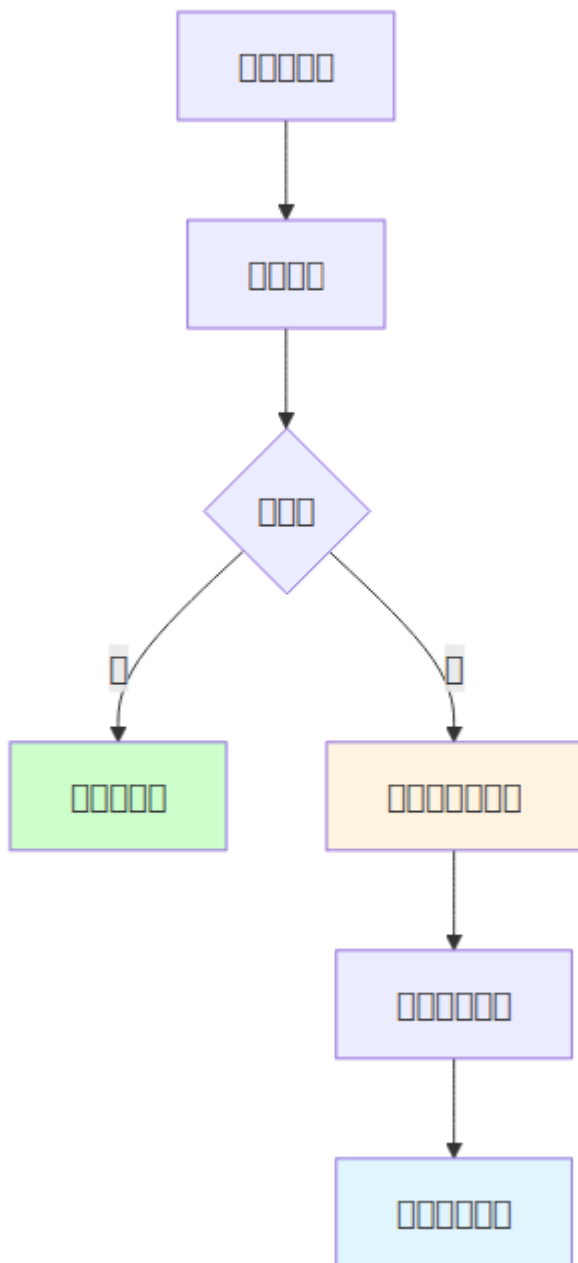
購買禮券

\$1 購買

購買\$10

購買\$11

OmniCRM



API 호출

```
curl -X POST https://your-domain.com/api/payments/invoice \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_KEY" \
-d '{
  "customer_id": 123,
  "amount": 10.00,
  "metadata": {
    "addon_id": 456
  }
}'
```

API

```
{
  "success": true,
  "message": "成功",
  "data": {
    "customer_id": 123,
    "service_amount": 10.00,
    "routing_mode": "hybrid",
    "initial_balance": 1.00,
    "wallet_portion_used": 1.00, // 钱包$1
    "card_portion_used": 9.00,  // 卡$9
    "charged_amount": 9.00,     // ← 卡
    "wallet_credited": 9.00,    // ← 钱包
    "wallet_debited": 10.00,    // ← 钱包
    "final_balance": 0.00
  }
}
```

2 * \$50 + \$30

\$50 + \$30

顧客\$50

店員
 $\$30 - \$50 = \$0$

店員

顧客

店員

顧客-\$30

顧客\$20
店員\$0
顧客\$30

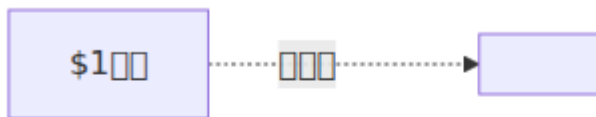
API

```
{
  "success": true,
  "data": {
    "service_amount": 30.00,
    "initial_balance": 50.00,
    "charged_amount": 0, // ← 0000
    "wallet_debited": 30.00,
    "final_balance": 20.00,
    "payment_method_used": false // ← 0000
  }
}
```

3

□□□□□□\$500□□□□□□□□\$150□

```
1 POST /api/payments/authorize/hold
```

**API**

```
{
  "authorization_id": 301,
  "amount": 500.00,
  "status": "authorized",
  "wallet_balance": 150.00,
  "wallet_to_use": 150.00,
  "card_amount": 350.00,
  "message": "□□□$350□□□□□□□□□□□□□□$500□□□□"
}
```

2 POST /api/payments/capture/301

□□□□

□□□□\$350

□□□□+\$350
□□□\$500

□□□□-\$500
□□□□

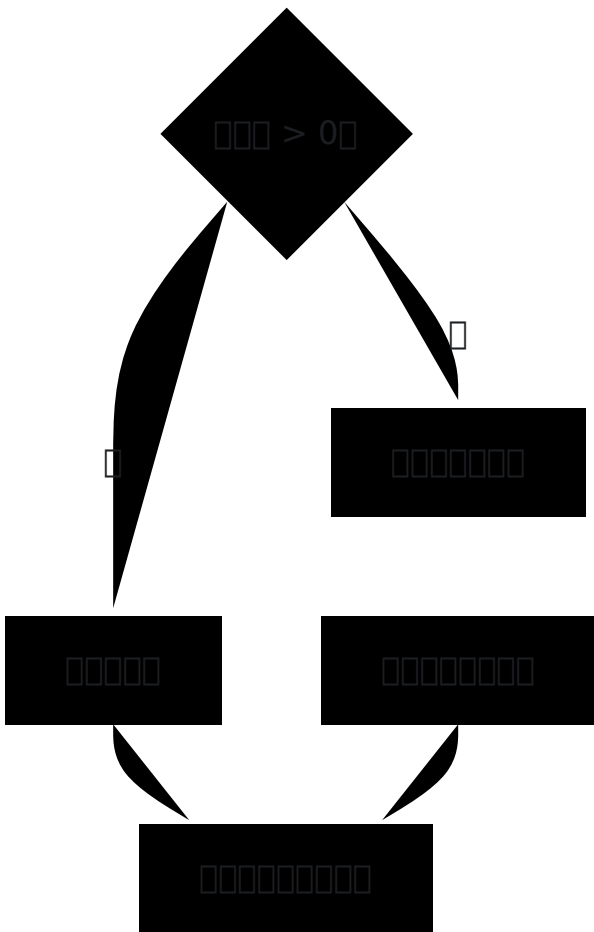
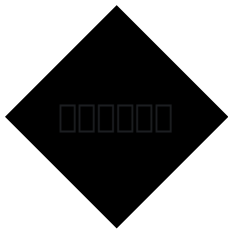
□□□□\$0
□□□□\$350

□□□□□

- □ □□□□□□□\$350□□□□\$500
- □ □□□□□□□□□□
- □ □□□□□□□□
- □ □□□□□□□

□□□□□

□□□□□□□□□□□



00

00000

1. 0000000000
2. 00000000 - 00000
3. 0000 > 00000000000
4. 0000000000
5. 0000000000
6. 000000000000000000

0000000

00000000000000000000

0000 - 00000000000000

```
curl -X POST https://your-domain.com/api/payments/charge \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer YOUR_API_KEY" \  
-d '{  
  "customer_id": 123,  
  "amount": 100.00,  
  "metadata": {  
    "routing_mode": "bypass"  
  }  
'
```

000000\$10000000\$10000000\$100000\$00

0000000000000000/000000000000

00000

1. 00000000000000000000000000

2. 環境変数からAPIキーを読み込む
3. 環境変数からAPIキーを読み込む
4. 環境変数からAPIキーを読み込む
5. 環境変数からAPIキーを読み込む
6. 環境変数からAPIキーを読み込む
7. 環境変数からAPIキーを読み込む7回
8. 環境変数からAPIキーを読み込む

Playbookの書き方

Ansible playbookの書き方

1. 環境変数/変数 - try/catchでエラーを処理
2. 変数**authorization_id** - 環境変数/変数
3. 変数**API** - 環境変数/変数
4. 環境変数 - 環境変数2回
5. 環境変数 - 環境変数からAPIキーを読み込む

環境変数からAPIキーを読み込む**Playbook**の書き方

環境変数

APIキーを読み込むAPIキーを読み込むAuthorization

Authorization: Bearer YOUR_API_KEY

環境変数からAPIキーを読み込む

- 000000 → 0000000
 - 000000 → 000Playbook00
-



API API API API

Transactions </payments_transaction>

Mailjet <integrations_mailjet> PDF

環境構築

OmniCRM の HTML テンプレートは Jinja2 を利用して生成されます。

テンプレート

テンプレートは `OmniCRM-API/invoice_templates/`

にあります。

- `norfone_invoice_template.html` - ノーフォンのテンプレート
- `cifi_invoice_template.html` - CIFのテンプレート

例

テンプレートは `OmniCRM-API/crm_config.yaml` に定義

```
invoice:
  template_filename: 'norfone_invoice_template.html'
```

インストール

OmniCRM の Jinja2 テンプレート

□□□□□

- `{{ invoice_number }}` - □□□□ ID□□□□ `INV-2025-001234`□
- `{{ date }}` - □□□□□□□□ISO □□□ `2025-01-10T12:00:00`□
- `{{ due_date }}` - □□□□□□□□□□ `2025-02-10`□
- `{{ start_date }}` - □□□□□□□□
- `{{ end_date }}` - □□□□□□□□
- `{{ total_amount }}` - □□□□□□□□□□
- `{{ total_tax }}` - □□□□□□□□□□□□□□

□□□□□

- `{{ client.name }}` - □□□□□□□□□□
- `{{ client.address.address_line_1 }}` - □□□ 1
- `{{ client.address.address_line_2 }}` - □□□ 2
- `{{ client.address.city }}` - □□
- `{{ client.address.state }}` - □/□
- `{{ client.address.zip_code }}` - □□□□
- `{{ client.address.country }}` - □□

□□□□□

□□□□□□□□□□□□□□

```
{% for sub_transaction in transactions %}
  <tr>
    <td>{{ sub_transaction.transaction_id }}</td>
    <td>{{ sub_transaction.created.split("T")[0] }}</td>
    <td>{{ sub_transaction.title }}</td>
    <td>{{ sub_transaction.description }}</td>
    <td>${{ "%.2f"|format(sub_transaction.retail_cost) }}</td>
  </tr>
{% endfor %}
```

□□□□□

- `sub_transaction.transaction_id` - □□ ID

- `sub_transaction.created` - 日期/时间
- `sub_transaction.title` - 标题
- `sub_transaction.description` - 描述
- `sub_transaction.retail_cost` - 零售成本
- `sub_transaction.tax_percentage` - 税率 10 或 10%
- `sub_transaction.tax_amount` - 税额

模板代码

```
<td>
    {% if sub_transaction.tax_amount and
sub_transaction.tax_amount > 0 %}
        ${{ "%.2f"|format(sub_transaction.tax_amount) }} ({{
sub_transaction.tax_percentage }}%)
    {% else %}
        -
    {% endif %}
</td>
```

模板文件

第 1 步

```
cd OmniCRM-API/invoice_templates/
cp norfone_invoice_template.html
your_company_invoice_template.html
```

第 2 步 **HTML/CSS**

将 `your_company_invoice_template.html` 复制到

模板文件

1. 模板文件

```
<!-- 公司标志 URL -->
![公司标志](https://yourcompany.com/logo.png)

<!-- 主标题 -->
<h1>欢迎来到我们的网站</h1>
```

2. 样式

```
<style>
  /* 导航栏 */
  .navbar {
    background: linear-gradient(to bottom right, #your-color-1, #your-color-2);
  }

  /* 表格 */
  .table thead th {
    background-color: #your-brand-color !important;
    color: white !important;
  }

  /* 按钮 */
  .btn-primary {
    background-color: #your-brand-color;
  }
</style>
```

3. 页脚

```
<footer>
  <p>公司地址</p>
  <p>123 街道, 城市, 国家</p>
  <p>电话: +1-555-123-4567 | 邮箱: billing@yourcompany.com</p>
  <p>ABN/税号: 12345678900</p>
</footer>
```

4. 其他

```
<div class="payment-info">
  <h3>결제정보</h3>
  <p><strong>결제 URL</strong> | https://yourcompany.com/pay |</p>
  <p><strong>계좌입금 정보</strong></p>
  <ul>
    <li>회사명: Your Company Ltd</li>
    <li>BSB: 123-456</li>
    <li>계좌번호: 987654321</li>
    <li>계좌명: {{ invoice_number }}</li>
  </ul>
</div>
```

5. 약관

```
<div class="terms">
  <h4>약관</h4>
  <p>결제 후 30일 이내에 결제</p>
  <p>결제 수수료 2%</p>
  <p>문의: billing@yourcompany.com</p>
</div>
```

3. 설정

OmniCRM-API/crm_config.yaml

```
invoice:
  template_filename: 'your_company_invoice_template.html'
```

4. API

```
cd OmniCRM-API
sudo systemctl restart omnicrm-api
```

5. 테스트

1. 테스트
2. 테스트

3. PDF
- 4.

Jinja2

```
{% if total_amount > 1000 %}  
  <div class="high-value-notice">  
    <p><strong></strong>  - </p>  
  </div>  
{% endif %}  
  
{% if client.address.country == "Australia" %}  
  <p> GST$ {{ "%.2f"|format(total_amount * 0.10) }}</p>  
{% endif %}
```

```
invoice_template_en.html  
invoice_template_es.html  
invoice_template_fr.html
```

```

<!-- 结账金额 -->
<tr>
    <td colspan="4" class="text-right"><strong>总计</strong></td>
    <td>${{ "%.2f"|format(total_amount) }}</td>
</tr>
<tr>
    <td colspan="4" class="text-right"><strong>税金</strong></td>
    <td>${{ "%.2f"|format(total_tax) }}</td>
</tr>
<tr>
    <td colspan="4" class="text-right"><strong>总计</strong></td>
    <td>${{ "%.2f"|format(total_amount + total_tax) }}</td>
</tr>

```

在 `total_tax` 模板变量中定义 `tax_amount` 模板变量
`tax_percentage` 模板变量 `tax_percentage` 为 0%

模板变量

模板变量

```

<div class="qr-payment">
    ![[二维码]](https://api.qrserver.com/v1/create-qr-code/?
    size=150x150&data={{ payment_url }})
    <p>二维码</p>
</div>

```

PDF 模板

OmniCRM 使用 **WeasyPrint** 将 HTML 模板转换为 PDF 模板

使用 **CSS**

- 使用 CSS 2.1 规范
- 使用 CSS3 的 flexbox 布局
- 使用 `@font-face` 定义字体

模板

- JavaScript
- CSS 排版排版排版
- 排版
- 排版 CSS 排版

排版排版排版

```
@page {  
    size: A4;  
    margin: 1cm;  
}  
  
body {  
    font-family: Arial, sans-serif;  
    font-size: 10pt;  
}
```

排版排版排版

```
@media print {  
    .no-print {  
        display: none;  
    }  
  
    .page-break {  
        page-break-after: always;  
    }  
}
```

排版排版

```
.table {
  table-layout: fixed;
  width: 100%;
}

.table th, .table td {
  word-wrap: break-word;
  padding: 4px;
}
```

□□□□□

□□□□□□□□□□□□□□□□□□□□

```
@font-face {
  font-family: 'YourFont';
  src: url('https://yourcompany.com/fonts/yourfont.woff2')
  format('woff2');
}

body {
  font-family: 'YourFont', Arial, sans-serif;
}
```

□□□□□□□

□□□□□

1. □□□□□
 - □□□□□□
 - □□□□□□□□
 - □□□□□□□□□□
 - □□□□□□
 - □□□□□□
2. □□□□□□
 - □□□□□□□□
 - □□□□□□□□

- ### 3. PDF ☐☐☐

4.

5.

- □ □ □ □ □ □ □ □ □

□□□□ API □□□□□□

□ □ □ □ □

□ □ □ □ □ □

- PDF 00000

- 了解 了解 CSS 的 历史
- 了解 了解 CSS 2.1 的 规范 WeasyPrint 的 CSS

URL

- URL 必須
- HTTPS URL

Table

- Table 必須
- Table 必須 `table-layout: fixed`

Font

- Font 必須
- Font 必須 Arial/Times New Roman

PDF

- HTML WeasyPrint
- HTML WeasyPrint

PDF

PDF OmniCRM PDF PDF PDF

PDF

1. PDF
 - PDF
 - PDF Base64
 - PDF SHA256
 - Invoice_PDF_Cache
 - ID
 - PDF Base64
 -
 -
 -

2. 查詢 - 查詢所有發票
 - 查詢 invoice_id 查詢 PDF
 - 查詢所有發票的 PDF
 - 查詢 `last_accessed` 查詢所有發票
3. 查詢 - 查詢所有發票的 PDF 查詢
 - 查詢所有發票/查詢所有發票
 - 查詢所有發票
 - 查詢所有發票

查詢

- 查詢 - 查詢所有發票的 PDF 查詢
- 查詢 - 查詢所有發票的 PDF 查詢
- 查詢 - 查詢 PDF 查詢 CPU 查詢
- 查詢 - 查詢所有發票的 PDF 查詢

查詢

查詢 PDF 查詢所有發票的 PDF 查詢

- 查詢所有發票 90 查詢
- 查詢所有發票 30 查詢
- 查詢所有發票的 PDF 查詢

API 查詢

查詢 API 查詢 UI 查詢

- 查詢所有發票的 PDF 查詢
- 查詢所有發票的 PDF
- 查詢所有發票的 PDF

查詢所有發票的 PDF 查詢

```
-- 查詢所有發票的 PDF 查詢 MySQL 查詢
DELETE FROM Invoice_PDF_Cache;
```

```
crm_config.yaml
```

--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□□□

1. 0000 0000 → **[00000000]**
2. 00 00 0000
3. 0000000000000000

□ □ □ □ ◆ ? ◆ ? □ □

1. $\square\square\square\square \rightarrow \square\square\square\square\square\square\square\square$
2. $\square\square\square\square\square\square\square\square\square\square$

--	--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□□□

- □□□ - □□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□ - □□□□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□

□□□□□

- 1,000 000000 "k" 000000\$1.5k
- 1,000,000 000000 "M" 000000\$2.3M
- 1,000,000,000 000000 "B" 000000\$1.1B

□□□□□

- “ ” “ ”
-
-
-

--	--	--	--

□□□□□□□□□□□□□□□□

□□□□

- **ID** - ID
- - /
- - - "N/A"
- -
- -
- -
- -
- -

- **↓ ()** - PDF
- **()** -
- **()** -
- **ⓧ ()** -
- **()** - Stripe Stripe

" + " "

- [illegible]

□ □ □ □ □

- ✓ (0000) - 000000000000
- × (00 X) - 000000000000
- 00 - 0000000000
- 0000 - 000000
- 00000000000000000000
- 00000000/0000000000

□ □ □ □ □ □

1. 00000000000000000000
2. 000000000000/000000
3. 00000000000000000000
4. 000000000000000000
5. 0000 PDF 000
6. 000000000000 invoice_id 00000
7. 0000000000000000000000
8. 0000000 "000" 0000

□□□□□

[illegible][illegible][illegible]

- PDF - 文件格式

PDF

PDF (↓) 文件格式 "PDF" 文件 PDF

PDF

- PDF 文件
- PDF 文件
- PDF 文件 Invoice_01234.pdf
- PDF 文件

PDF 文件

- PDF - 文件 PDF 文件 2-3 文件
- PDF - 文件
- PDF - 文件

PDF

- PDF - 文件 API 文件
- PDF 文件 - 文件
- PDF - 文件

PDF

PDF (↓) 文件

목차

1. 결제시스템의 종류
2. 결제시스템의 구성요소
3. 결제시스템
 - **Stripe** 결제 - 결제시스템의 종류
 - 결제 - 결제시스템의 종류
 - 결제 - 결제시스템의 종류
 - **POS** 결제 - 결제시스템의 종류
 - 결제 - 결제시스템의 종류
4. 결제시스템 Stripe
 - 결제시스템의 종류
 - 결제시스템
 - 결제시스템
5. 결제시스템의 종류
 - 결제시스템의 종류
6. 결제 "결제" 결제
7. 결제시스템
 - **Stripe** - 결제 Stripe API 결제
 - 결제 - 결제시스템의 종류
8. 결제시스템 "결제"
9. 결제시스템

2. 顧客の請求書 PDF を生成する
3. 以下 Mailjet <integrations_mailjet> の `api_crmCommunicationCustomerInvoice` を呼び出す
4. 返されるデータ
 - 請求書 PDF
 - 請求書 ID
 - 請求書の日付
 - 請求書の総額
 - 請求書の URL
 - 請求書の支払 URL
5. 請求書のデータをメールで送信する

準備作業

請求書のデータをメールで送信する "メール" を送信するための準備作業を行います。

必要なデータ

- `{{ var:customer_name }}` - 顧客名
- `{{ var:invoice_number }}` - 請求書 ID
- `{{ var:invoice_date }}` - 請求書の日付
- `{{ var:due_date }}` - 請求書の支払期限
- `{{ var:total_amount }}` - 請求書の総額
- `{{ var:invoice_url }}` - 請求書の PDF の URL
- `{{ var:pay_url }}` - 請求書の支払 URL

必要な設定

- 請求書のデータ - 以下 `crm_config.yaml` の Mailjet API の設定
- 請求書のデータ - 請求書のデータを取得するための API
- **PDF** のデータ - 請求書の PDF を生成するための API

実行

以下コマンド () を実行する

□□□

- □□□□□ □□□
- □□□□□□□□□□□□□□□□

□□□□□

1. □□□□□□□□□□□□
2. □□□□□□ (□)
3. □□□□□□□□

□□□□□□

- □□□□□ `void = true`
- □□□□□□□□□□□□ `invoice_id` □□□ `null`□
- □□□□□□ "□□□"
- □□□□□□□□□□□□
- □□□□□□□□ "□□□" □□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□ "□□" □□□□□□

□□□□□

- □□□□□□□□
- □□ = "□□□□□□□□□□□□□□□□□□□□□□"

- `id = "xxxxxxxxxxxx"xxxxxxxxxxxx`

退款

退款函数 (`refund`) 退款

参数

- 退款金额
- 退款来源 **Stripe** 支付
- 退款参考 `payment_reference` (Stripe 支付 ID)
- 退款描述

步骤

1. 创建 Stripe 支付
2. 退款 (`refund`)
3. 退款成功

4. 返回 "成功"

5. 退款 Stripe 支付

- 调用 Stripe API 退款
- 传入 Stripe 支付 ID
- 退款参考 `refund_reference`

6. 退款成功 "成功"

7. 決済決済

◆◆◆決済決済決済

- 決済決済決済決済決済
- 決済 "決済"
- 決済決済
- 決済決済決済決済3-7 決済
- Stripe 決済決済決済

決済

- 決済決済決済POS 決済決済決済決済決済
- 決済決済決済決済決済
- 決済

決済決済決済

決済

決済決済決済決済決済

- 決済 ID
- 決済
- 決済決済決済決済

決済

決済決済決済決済

決済

- 決済 - 決済決済決済
- 決済 - 決済決済決済決済

決済

7. ☐
 - ☐ ☐
 - ☐ ☐
 - ☐ ☐
8. ☐ " ☐ " ☐ " ☐ 15 ☐ "
9. ☐ ☐
10. ☐ ☐
11. ☐ ☐ " ☐ " ☐
12. ☐ PDF ☐

☐ 2 ☐

1. ☐ + ☐
2. ☐
3. ☐ ☐ ☐
 - ☐ " + ☐ "
 - ☐ " ☐ - 2025 ☐ 1 ☐ "
 - ☐ / ☐ 1 ☐ 1 ☐ 31 ☐
 - ☐ 2 ☐ 15 ☐
 - ☐
 - ☐ **X** ☐
 - ☐ **X** ☐
 - ☐
 - ☐
 - ☐ " ☐ " ☐
4. ☐
 - ☐ " + ☐ "
 - ☐ " ☐ - 2025 ☐ 1 ☐ "
 - ☐ / ☐ 1 ☐ 1 ☐ 31 ☐
 - ☐
 - ☐
 - ☐ **X** ☐
 - ☐
 - ☐ " ☐ "

5. 0000000000000000
 - 0000 "+ 0000000"
 - 000"00000 - 2025 0 1 0"
 - 0000000000000000
 - 00 "00" 000000
 - 00 "0000"
6. 00000000000000000000

0000 **3**00000000000000

1. 0000000
2. 00000 0000 000
3. 00 "+ 0000000"
4. 0000000000
5. 0000000 20 000
6. 0000000000000000000000
7. 00000000
 - 00000000000"0000000"0
 - 00 **X** 0000000
 - 000000000000"00000"0
 - 00 **X** 0000000
 - 00000000"0000 18 000"
 - 0000000000
8. 00000000000000000000
9. 00 "00000**18**0"
10. 0000000000000000
11. 00/0000000000000000000000

0000 **4**000000000000

1. 000000000000
2. 00000000000000
3. 000000000000
 - 00000000000000

- 請 X 刪除
- 刪除
- 請 X 刪除

4. 刪除
5. 刪除
6. 請 "刪除" 刪除
7. 刪除/刪除
8. 刪除

請 5 刪除

1. 刪除
2. 刪除
3. 刪除 請 刪除
4. 請 "+ 刪除"
5. 刪除
 - 刪除
 - 刪除
 - 刪除 + 7
 - 刪除“刪除”
6. 刪除
7. 刪除
8. 請 X 刪除
9. 請 "刪除"
10. 刪除
11. 刪除 Stripe 刪除

請 6 刪除

1. 刪除
2. 刪除
3. 刪除 刪除
4. 刪除
 - 請 ID 刪除

- 0000000000
- 0000000000
- 0000000000

5. 00000000000000
6. 000000 PDF00000000 "00 **PDF**" 00
7. 000000000000000000 "000000" 00
8. 0000000000

0000 **7**0000000000

1. 0000000000
2. 0000000000000000
3. 000000000000000000
4. 00000000
5. 0000000000000000
6. 000000000 "00" 00
7. 00000
8. 00000000000000
9. 000000000000000000000000
10. 000000000000000000
 - 000000000000000000000000
 - 0000000000
11. 000000000000000000000000

0000 **8**0000000000

1. 0000000000000000
2. 000000 00 0000
3. 000000000000
4. 000000000000000000
5. 0000000000
6. 0000000000 "00000" 00
7. 00 "00" 00000
8. 000000 "00000 2025-01-15"

9. `int "aaaa"`
10. `aaaaaaaaaaaa "aaa"`
11. `aaaaaaaaaaaa`
12. `aaaaaaaaaaaa "aaa"`

□□□□ **9**□□□□□□□□

1. 创建 Stripe 账户
2. 创建 Stripe 商户
3. 创建 Stripe 产品
4. 创建 Stripe 价格
5. 创建 Stripe 支付链接
6. 创建 Stripe 支付按钮
7. 创建 Stripe 支付表单
8. 创建 Stripe 支付网关
9. 创建 Stripe 支付网关
10. 创建 Stripe 支付网关
11. 创建 Stripe 支付网关

□ □ □ □

$$\square\square\square\square\square\square - \square\square\square\square\square$$

- [illegible]

PDF 文件

- 目前 市面上有 WeasyPrint 這個方案
- 目前 市面上 HTML 轉 PDF 的方案 基本上 API 都

Stripe

- 如何 集成 Stripe API 包
- 如何 集成 Stripe 库

環境構築

- 必要パッケージ
- 必要ライブラリ/フレームワーク Stripe 連携ライブラリ

設定

- 必要 Mailjet API キー取得
- 必要 `crm_config.yaml` 必要 Mailjet 連携設定

実装

- 必要 決済/POS/請求 Stripe連携
- 必要 請求 Stripe 連携

PDF 生成

- 必要 PDF 生成
- 必要 請求 PDF 生成 `DELETE FROM Invoice_PDF_Cache WHERE invoice_id = X;`

テスト

- 必要 テスト環境構築
- 必要 テスト 実行

デモ

- 必要 デモ環境構築
- 必要 デモ 実行

運用

- 必要 運用環境構築
- 必要 運用 実行

その他

- 必要 その他
- 必要 その他



□ □

□□□□□□□□□□□□□□□□“□□”□□

[illegible]

□ □

□ □

--	--	--	--

□ □

□ □

[illegible][illegible]

```
00000000 00 <payments_invoices>000000000000
```

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □

1. □□□ □□ → **[□□□□]**
2. □□ □□ □□
3. □□□□□□□□□□□□

□ □ □ □ □ □ □

1. $\square\square\square\square \rightarrow \square\square\square\square\square\square\square\square$
2. $\square\square\square\square\square\square\square\square\square\square$

--	--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□□□

- 0000 - 0000000000000000
- 000000 - 00000000000000
- 000000 - 00000000000000
- 000000 - 00000000000000

5/5

- 1,000 “k” \$1.5k
- 1,000,000 “M” \$2.3M
- 1,000,000,000 “B” \$1.1B

--	--	--	--

□□□□□□□□□□□□□□□□

□□□□□□□/□□□□

□□□□□□□□□□□□

- □□□□ - □□□□□□□
- □□□ - □□□□□□□/□□□□□
- □□□□□□□ - □□□□□□□□□□□
- □□□□ - □□□□□□□
- □□ - □□□□□□□
- □□ - □□□□□□□

□□□□□□□

□□ "+ □□□□" □□□□□□□□□□

□□□□□□□□□

□□□□□□□/□□□□

အခြေခံ

- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံ - အခြေခံအားဖြင့်/အခြေခံ

အခြေခံ

- အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံအားဖြင့်/အခြေခံ
- အခြေခံအားဖြင့်/အခြေခံ

အခြေခံ

1. အခြေခံအားဖြင့်/အခြေခံ
2. အခြေခံအားဖြင့်/အခြေခံ

3. 請將“ ”填上
4. 請將填上
5. 請將填上

請將填上

請將

請將填上

- ID
-
-
- ID

請將

請將填上

請將填上

- -
- -

請將填上

- -
- -

請將

請將填上

- **ID** - ID/
- -
- -

- 00 - 000000000000/0000
- 00 - 0000ID00

0000000000000000 ↔ 00000

00000

000000000000 000000000000

0000

- 00000000
- 000000000000000000000000

000000

1. 0000000000
2. 00000000 : 0
3. 00 "0000"
4. 00000000

0000000

- 000000 void = true
- 0000000000000000

- 10% of \$50.00
- $\$50.00 \times 0.10 = \5.00
- \$5.00 (10%)

Nil/

- 0
- 0%
- “-”

-
-
-
- “”

- 00000000 (-)

00000000

- 00000000
- 000000000000000000000000
- 00ID0000000000000000
- 00000000 (✓)
- 0000

000000

0000000000000000

1. 000000000000000000000000
2. 0000000000000000
3. 0000000000000000
4. 000 invoice_id 000000
5. 00000000“0000”

000 payments_invoices 000000000000000000000000

00000000

00000 **1**0000000000000000

1. 000000“00000020”
2. 00000000£100000
3. 000000 00 00
4. 00 "+ 000000"
5. 00 00 00000
6. 00 00000 00000
7. 000000“00000000”
8. 000000“0108-90000000”
9. 00000000010.00

10. 0000000000000000
11. 00 "0000"
12. 000-£10.00000000
13. 0000000000000000

0000 2000000000

1. 000000000000
2. 000000£75000000
3. 000000 00 00
4. 00 "+ 0000"
5. 00 00 0000
6. 000000"0000"
7. 000000"00000000000000"
8. 0000000075.00
9. 0000000045.000000000000
10. 0000000000
11. 00000000
12. 00 "0000"
13. 00000000000000
14. 000000000000

0000 3000000000

1. 000000000000
2. 0000000000
3. 0000000000000000 : 0
4. 00 "0000"
5. 0000000000
6. 0000000000
7. 000000000000

□□□□

- `payments_invoices` - □□□□□□
- `payments_process` - □□□□□□
- `basics_payment` - □□□□□□
- `csa_activity_log` - □□□□□□□□□□

Playbooks

Ansible playbooks

API

OmniCRM Ansible Playbooks

- 1.
- 2.

Playbooks

retail_cost wholesale_cost retail_setup_cost

Ansible playbook

- api_response_product.json.retail_cost
-
-
-
-
-
-
-

API Test Script

```
# 零售成本从 $99.00 降至 $79.00 的测试
- name: 零售成本测试
  set_fact:
    charge_amount: 79.00

- name: 发起支付
  uri:
    url: "http://localhost:5000/crm/payments/authorize/hold"
    method: POST
    body_format: json
    body:
      customer_id: "{{ customer_id | int }}"
      amount: "{{ charge_amount | float }}" # 零售成本
  register: retail_cost
  # ...
```

API Test Script

```
# 客户注册和定价测试
- name: 客户注册
  uri:
    url: "http://localhost:5000/crm/customer/{{ customer_id }}"
    # ...
  register: customer_info

- name: 获取价格
  set_fact:
    base_price: "{{ api_response_product.json.retail_cost | float }}"
    discount_percent: >-
      {% if customer_info.json.tier == 'gold' %}20
      {% elif customer_info.json.tier == 'silver' %}10
      {% else %}0{% endif %}

- name: 计算最终价格
  set_fact:
    final_price: "{{ (base_price | float) * (1 - (discount_percent | float / 100)) | round(2) }}"
```

ansible-playbook

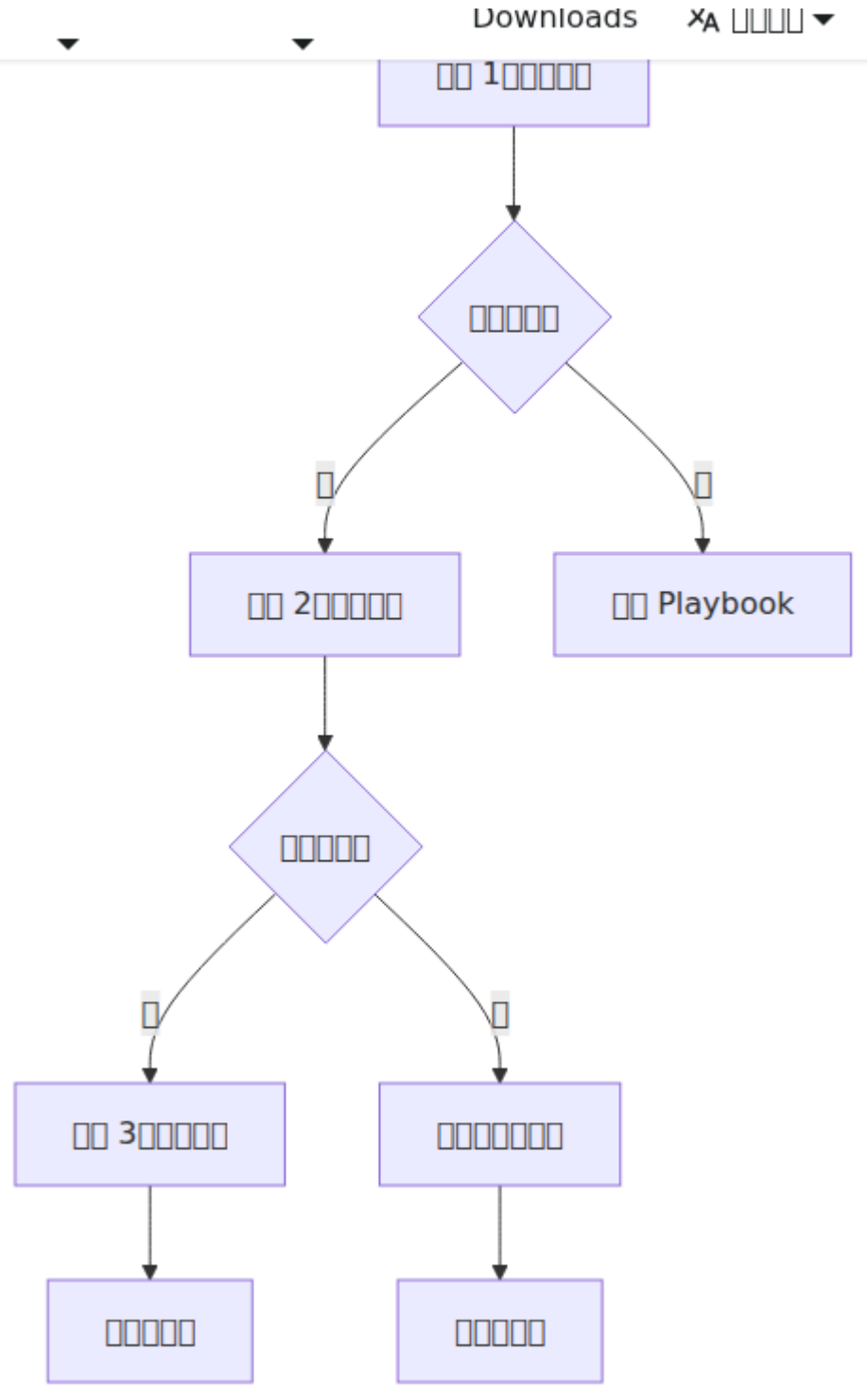
```
# 根据是否是 wholesale_cost 还是 retail_cost
- name: 设置事实
  set_fact:
    charge_amount: >-
      {% if customer_info.json.is_wholesale_partner %}
      {{ api_response_product.json.wholesale_cost | float }}
      {% else %}
      {{ api_response_product.json.retail_cost | float }}
      {% endif %}
```

ansible-playbook 是 ansible 的入口，它负责执行 playbook 中的任务。CRM 系统/数据库
ansible-playbooks 是 ansible 的配置文件

ansible-playbook ? ? ?

ansible-playbook 是 ansible 的入口，它负责执行 playbook 中的任务。

□□□□



□□□□

□□□□□□□□□□

□□ **1**□□□ - □□□□□□□□□□

□□ **2**□□□ - □□ OCS □□/□□□□

00 **3**000 - 0000000000

00 - 0000000000

0000

0000 play_topup_charge_then_action.yaml 000000

```

- name: Play Topup - Charge card then action the Topup
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: 创建 crm_config 文件
      ansible.builtin.include_vars:
        file: "../crm_config.yaml"
        name: crm_config

    # 获取 token
    - name: 获取 CRM API 返回 token
      uri:
        url: "http://localhost:5000/crm/product/product_id/{{
product_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        validate_certs: no
        register: api_response_product

    - name: 获取 CRM API 返回服务
      uri:
        url: "http://localhost:5000/crm/service/service_id/{{
service_id }}"
        method: GET
        headers:
          Authorization: "Bearer {{ access_token }}"
        return_content: yes
        validate_certs: no
        register: api_response_service

    - name: 设置事实
      set_fact:
        service_uuid: "{{ api_response_service.json.service_uuid
}}"

        customer_id: "{{ api_response_service.json.customer_id }}"
        package_name: "{{ api_response_product.json.product_name
}}"

        monthly_cost: "{{ api_response_product.json.retail_cost
}}"

```

```

        wholesale_cost: "{{
api_response_product.json.wholesale_cost }}"

# =====
- name: 获取支付方法
  uri:
    url: "http://localhost:5000/crm/payments/methods?
customer_id={{ customer_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
    return_content: yes
    validate_certs: no
  register: api_response_payment_methods

- name: 根据支付方法id 获取支付方法详情
  set_fact:
    payment_method_id: "{{ api_response_payment_methods.json |
json_query(query) }}"
  vars:
    query: "data[?is_default==`true`].payment_method_id | [0]"

# =====
# =====

- name: "调用 1 获取支付单"
  uri:
    url: "http://localhost:5000/crm/payments/authorize/hold"
    method: POST
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": "{{ customer_id | int }}",
        "amount": "{{ monthly_cost | float }}",
        "currency": "{{ crm_config.currency | default('AUD')
}}",
        "payment_method_id": "{{ payment_method_id | int }}",
        "metadata": {
          "description": "{{ package_name }} on {{
api_response_service.json.service_name }}"

```

```

        "service_id": "{{
api_response_service.json.service_id | int }}",
        "site_id": "{{ api_response_service.json.site_id |
int }}",
        "product_id": "{{ product_id }}",
        "user_id": "{{ (initiating_user | int) if
(initiating_user is defined and initiating_user is not none) else
omit }}",
        "title": "{{ package_name }}",
        "wholesale_cost": "{{ wholesale_cost | float }}",
        "invoice": true,
        "contract_days": "{{ days | int }}",
        "send_email": true
    }
}
return_content: yes
register: api_response_authorization

```

- name: 授权成功
 - assert:
 - that:
 - api_response_authorization.status == 200
 - api_response_authorization.json.success == true

```

- name: 设置 authorization_id 为 9999/99
  set_fact:
    authorization_id: "{{
api_response_authorization.json.data.authorization_id }}"

```

调用 200CS 接口获取 CGRateS 数据

- block:
 - name: "调用 200CS CGRateS 接口"
 - uri:
 - url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
 - method: POST
 - body_format: json
 - headers:
 - Content-Type: "application/json"
 - Authorization: "Bearer {{ access_token }}"
 - body:
 - {
 - "id": "{{ 999999999 | random }}"
 - "method": "APIerSv1.ExecuteAction",
 - "params": [{


```

        "Tenant": "{{ crm_config.ocs.ocsTenant }}",
        "Account": "{{ service_uuid }}",
        "ActionsId": "{{ cgr_action_name }}"
    }]
}
status_code: 200
register: action_execute_response

- name: 成功 OK
  assert:
    that:
      - action_execute_response.status == 200
      - action_execute_response.json.result == "OK"

# 3番目のテスト - 支払いのキャプチャ
- name: "3番目のテスト"
  uri:
    url: "http://localhost:5000/crm/payments/capture/{{
authorization_id }}"
    method: POST
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "metadata": {
          "provisioning_status": "success",
          "cgr_action": "{{ cgr_action_name }}"
        }
      }
    return_content: yes
  register: api_response_capture

- name: 成功
  assert:
    that:
      - api_response_capture.status == 200
      - api_response_capture.json.success == true

rescue:
  # エラーハンドリング
  - name: "エラーハンドリング"
    uri:

```

```

url: "http://localhost:5000/crm/payments/release/{{
authorization_id }}"
method: POST
headers:
  Content-Type: "application/json"
  Authorization: "Bearer {{ access_token }}"
body_format: json
body:
  {
    "metadata": {
      "release_reason": "provisioning_failed",
      "cgr_action": "{{ cgr_action_name }}"
    }
  }
return_content: yes
register: api_response_release

- name: ansible playbook
fail:
  msg: "OCS provisioning {{ authorization_id }} failed"

```

API

1. Provisioning failed for \$1 and \$10 payments
 2. \$9 payment provisioning failed

1. /

POST /crm/payments/authorize/hold

1. \$150 payment \$500 payment \$350 payment
 2.

```

{
  "customer_id": 123,
  "amount": 49.99,
  "currency": "AUD",
  "payment_method_id": 456,
  "metadata": {
    "description": "xxxxxxxx",
    "service_id": 789,
    "site_id": 12,
    "product_id": "34",
    "user_id": 5,
    "title": "xxxx",
    "wholesale_cost": 25.00,
    "invoice": true,
    "contract_days": 30,
    "send_email": true
  }
}

```

xxxxxx

- **customer_id**xxxx - xxxxxxxx ID
- **amount**xxxx - xxxxxxxxxxxxxxxx
- **currency**xxxx - xxxxxxxxxxxxxxxx
- **payment_method_id**xxxx - xxxxxxxx
- **metadata**xxxx - xxxxxxxxxxxx
 - **description** - xxxx
 - **service_id** - xxxxxxxx
 - **site_id** - xxxxxxxxxxxx
 - **product_id** - xxxxxxxx
 - **user_id** - xxxxxxxxxxxx
 - **title** - xxxx
 - **wholesale_cost** - xxxxxxxxxxxx
 - **invoice** - xxxx **true**xxxxxxxxxx
 - **contract_days** - xxxx
 - **send_email** - xxxx **true**xxxxxxxxxx

返回

```
{
  "success": true,
  "message": "充值成功",
  "data": {
    "authorization_id": 301,
    "vendor_authorization_id": "auth_xxxxx",
    "amount": 500.00,
    "currency": "USD",
    "status": "authorized",
    "wallet_balance": 150.00,
    "wallet_to_use": 150.00,
    "card_amount": 350.00,
    "message": "充值 $350 成功，当前余额 $500"
  }
}
```

说明

- `authorization_id` 充值授权ID
- `card_amount` 充值金额
- 充值成功后，系统会自动生成一条充值记录

2. 充值

接口 `POST /crm/payments/capture/{authorization_id}`

请求参数：`invoice: true` (是否生成发票)

返回

```
{
  "metadata": {
    "provisioning_status": "success",
    "cgr_action": "Action_Topup_Standard",
    "additional_info": "充值成功"
  }
}
```

####

```
{
  "success": true,
  "data": {
    "payment_id": "pay_xyz789",
    "transaction_id": 1234
  }
}
```

#####

1. #####
2. **###***#####
3. #####
4. ##### `invoice: true`
 - ##### = ##
 - #####
 - ##### = #####
 - #####
 - #####
5. #####
6. ## `send_email: true`#####

###\$500 ### \$150 #####

- #####\$350
- #####+\$350##### \$500
- #####-\$500#####
- #####\$0

#####

3.

`POST /crm/payments/release/{authorization_id}`

00000000000000000000000000000000/000000

0000

```
{
  "metadata": {
    "release_reason": "provisioning_failed",
    "error_details": "OCS 000000"
  }
}
```

000

```
{
  "success": true,
  "message": "000000"
}
```

000000

1. 0000000000000000
2. 0000000000000000
3. 0000000000000000
4. 00000
5. 00000000

00000000000000000000000000000000

00000000

00000000000000000000000000000000 API 00000000

00 **API** 0000

000 PUT /crm/transaction/

00 play_simple_service.yaml 0000

```

- name: [] API []
  uri:
    url: "http://localhost:5000/crm/transaction/"
    method: PUT
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json
    body:
      {
        "customer_id": {{ customer_id | int }},
        "service_id": {{ service_creation_response.json.service_id
| int }},
        "title": "{{ package_name }} - []",
        "description": "[ {{ package_comment }} []",
        "invoice_id": null,
        "wholesale_cost": {{
api_response_product.json.wholesale_setup_cost | float }},
        "retail_cost": "{{
api_response_product.json.retail_setup_cost | float }}"
      }
    return_content: yes
    register: api_response_transaction

- name: []
  assert:
    that:
      - api_response_transaction.status == 200

```

[]

- **customer_id**[] - [] ID
- **service_id**[] - [] ID
- **title**[] - []
- **description**[] - []
- **invoice_id**[] - [] ID [] null
- **wholesale_cost**[] - []
- **retail_cost**[] - []
- **site_id**[] - [] ID

- **tax_percentage**税率 - 税率

税率

- 税率
- 税率
- 税率
- 税率
- 税率

税率 税率 - 税率

税率

税率

- 税率
- 税率/税率
- 税率

税率

pro_rata_charge = (monthly_cost × days_remaining) / days_in_month

税率

税率 `playbook` 税率


```

# 00000000000000000000
# 000000 15 0000000000 1 00
# 00000000000000000000

- name: 000000000000
  command: "date +%d"
  register: current_day

- name: 000000000000
  command: "date -d 'last day of this month' +%d"
  register: days_in_month

- name: 000000000000
  command: "date -d 'last day of this month' +%Y-%m-%d"
  register: last_day_of_month

- name: 000000000000
  set_fact:
    days_remaining: "{{ (days_in_month.stdout | int) -
(current_day.stdout | int) + 1 }}"

- name: 000000000000
  set_fact:
    pro_rata_cost: "{{ ((monthly_cost | float) * (days_remaining |
float) / (days_in_month.stdout | float)) | round(2) }}"

- name: 000000000000
  debug:
    msg:
      - "000000${{ monthly_cost }}"
      - "000000${{ days_in_month.stdout }}"
      - "000000${{ days_remaining }}"
      - "00000000${{ pro_rata_cost }}"

# 00000000000000000000 pro_rata_cost
- name: "000000000000"
  uri:
    url: "http://localhost:5000/crm/payments/authorize/hold"
    method: POST
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json

```

```
body:
{
  "customer_id": "{{ customer_id | int }}",
  "amount": "{{ pro_rata_cost | float }}",
  "currency": "{{ crm_config.currency | default('AUD') }}",
  "payment_method_id": "{{ payment_method_id | int }}",
  "metadata": {
    "title": "{{ package_name }}" ({{ days_remaining }}
    )",
    "description": "{{ days_remaining }}/{{
days_in_month.stdout }}" {{ package_name }} ",
    "service_id": "{{ service_id | int }}",
    "invoice": true
  }
}
```

XXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

```

- name: 設定サービス開始日と請求日
  set_fact:
    service_start_date: "2024-01-15"
    next_billing_date: "2024-02-01"

- name: 日数計算のシェルスクリプトを実行
  shell: |
    echo $(( ( $(date -d "{{ next_billing_date }}" +%s) - $(date -d "{{ service_start_date }}" +%s) ) / 86400 ))
  register: days_until_billing

- name: 30日間の請求期間を設定
  set_fact:
    billing_period_days: 30

- name: 日割の料金を計算
  set_fact:
    pro_rata_cost: "{{ ((monthly_cost | float) * (days_until_billing.stdout | float) / (billing_period_days | float)) | round(2) }}"

- name: デバッグメッセージ
  debug:
    msg:
      - "サービス開始日: {{ service_start_date }}"
      - "次期請求日: {{ next_billing_date }}"
      - "経過日数: {{ days_until_billing.stdout }}"
      - "日割料金: ${{ pro_rata_cost }}"

```

例1: 17日間の請求

1. 条件設定

- サービス開始: 1月15日
- 月額料金: \$60.00
- 1月の総日数: 31
- 経過日数: 17日 (1月15日から1月31日まで)
- 日割料金: $\$60.00 \times 17 \div 31 = \32.90

2. 実行結果

- 0000000000 10 000
- 0000\$30/0
- 0000\$50/0
- 0000000030
- 0000021
- 000\$20/0
- 0000000 $\$20.00 \times 21 \div 30 = \14.00

0000

1. 00000/0000

000000000000/0000000000

```
- block:
  # 0000
  - name: 0000
    uri: ...

  # 0000000000
  - name: 0000
    uri: ...

rescue:
  # 0000000000000000
  - name: 000000
    uri: ...

  - name: 00 playbook
    fail:
      msg: "00000000000"
```

2. 0000 API 00

000000000000

```

- name: 注册
  uri: ...
  register: api_response_authorization

- name: 断言
  assert:
    that:
      - api_response_authorization.status == 200
      - api_response_authorization.json.success == true
    fail_msg: "断言失败{{ api_response_authorization.json }}"

```

3. 设置 ID

设置 authorization_id 为 123456789

```

- name: 设置 authorization_id 为 123456789
  set_fact:
    authorization_id: "{{
api_response_authorization.json.data.authorization_id }}"

```

4. 设置元数据

设置元数据为以下信息

```

metadata:
  description: "测试数据"
  service_id: "{{ service_id | int }}"
  product_id: "{{ product_id }}"
  user_id: "{{ initiating_user | int }}"
  title: "测试数据"
  wholesale_cost: "{{ wholesale_cost | float }}"
  invoice: true # 是否生成发票
  send_email: true # 是否发送邮件

```

5. 设置变量

设置变量为以下信息

```
- name: 計算最終コスト
  set_fact:
    final_cost: "{{ (base_cost | float * multiplier | float) |
round(2) }}"
```

6. 支払い方法の検証

支払い方法の検証

```
- name: 支払い方法の検証
  set_fact:
    payment_method_id: "{{ api_response_payment_methods.json |
json_query(query) }}"
  vars:
    query: "data[?is_default==`true`].payment_method_id | [0]"

- name: 支払い方法の検証
  assert:
    that:
      - payment_method_id is defined
      - payment_method_id != ""
      - payment_method_id != None
  fail_msg: "支払い方法 {{ customer_id }} が定義されていません"
```

支払い方法

1. 支払い方法の検証

支払い方法の検証

1. 支払い方法の検証
2. 支払い方法
3. OCS/CGRateS 支払い方法
4. CRM 支払い方法
5. 支払い方法
6. 支払い方法の検証

000 play_topup_charge_then_action.yaml 0000000

00 200000000000

0000000000000000

1. 0000
2. 000000
3. 000000000000000000
4. 000000000000

000 play_simple_service.yaml 00 202-232 00000000

00 300000/0000

000000000000

1. 000000
2. 00 CGRateS 00
3. 000000
4. 0000000000

00 4000 **ActionPlan** 000000

0000000000

1. 0000 *http_post 0000000000
2. 0000 *monthly 000 ActionPlan
3. 0 ActionPlan 00000
4. CGRateS 0000000000
5. 00 playbook 000000

□□□□

□□□□

□□□ □□□□□□□□

□□□□□

- □□□□□□□□□□
- □□□□
- □□□□□□
- □□ ID □□□

□□□□□ □□□□□□□□□□□□□□

□□□□□□□□□□□□

□□□ □□□□□□□□□□□□

□□□ □□□□□□□□□□ - □□□□□□□□□□□□□□

□□□□□

- □□□□□□□□□□ 7 □□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□

□□□□□□□□□□

□□□ □□□□□□□□□□□□

□□□ □□□□□□□□□□ invoice: true

□□□□□ □□□

- □□□□□□□□□□ invoice: true□□□

- 0000000000000000

0000000000

000 00000000000000

00000

- 0000000000000000/0000/00000
- 00000 0000000000
- 000000

00000

- 000000000000000000000000
- 0000000000000000
- 0000000000
- 000000000000

0000000000

0000

0000000000000000

1. 00000000 - 00000000/PayPal

```
- name: Refund
  uri:
    url: "http://localhost:5000/crm/payments/refund"
    method: POST
    headers:
      Content-Type: "application/json"
      Authorization: "Bearer {{ access_token }}"
    body_format: json
  body:
    {
      "transaction_id": "{{ vendor_transaction_id }}",
      "vendor": "stripe",
      "amount": "{{ refund_amount | float }}",
      "reason": "customer_request"
    }
```

2. Refund - Refund a payment made by a customer

Refund a payment made by a customer

- Refund
- Refund a payment made by a customer
- Refund

Refund a payment made by a customer

Refund

Refund a payment made by a customer

- Stripe ACH
- PayPal PayPal Refund

Refund a payment made by a customer Square Adyen Braintree

Refund

- Refund - Refund a payment made by a customer
- Refund - Stripe/PayPal Refund API

目次

目次 **Playbook** 目次

- `concepts_ansible.md` - ansible playbook 目次
- `concepts_provisioning.md` - 目次

目次

- **API** 目次 - API 目次
 - **API**
 - **API**
 -
 -
 -
- `payments_transaction.md` - 目次
- `payments_process.md` - 目次
- `basics_payment.md` - 目次

OmniCRM 安装指南

OmniCRM 安装指南

OmniCRM 是一个基于 **RBAC** 的 CRM 系统。安装过程中，您需要配置以下权限：

view_customer create_inventory

权限

RBAC 权限

- 权限 — 权限管理
- 权限 — 权限管理
- 权限 — 权限管理
- 权限 — “权限...” 权限管理/权限

权限

- 权限 — OmniCRM 权限
- 权限 — view_customer delete_product

-   —      *Admin* *Support* *Finance*
-  —                     

□□□□□□//JWT□API□□□□□□IP□□□□□□/□□□□□□//□□□□□□

--	--	--	--

OmniCRM

--	--	--	--

[illegible]

0000 - 0000000000000000AdminSupportFinance00000000CRM00000000
0000000000

```
0000 - 00 admin 000000000000000000000000000000000000000000
```

□□□□□□□ Omnitouch □□□□□□□□□□

API 与 Web UI 的交互

API 与 Web UI 的交互

Web UI

1. 数据输入 - 数据输出
2. “数据” - 数据

3. 数据输入
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
 - 数据 - 数据
4. “数据” - 数据
5. 数据 - 数据

数据

- TempP@ssw0rd! 비밀번호
-
-
-

API

POST /auth/users

admin

```
{
  "username": "john.smith",
  "email": "john.smith@company.com",
  "password": "TempP@ssw0rd!",
  "first_name": "John",
  "middle_name": "D",
  "last_name": "Smith",
  "phone_number": "+61412345678",
  "role": "Support"
}
```

```
{
  "id": 123,
  "username": "john.smith",
  "email": "john.smith@company.com",
  "first_name": "John",
  "last_name": "Smith",
  "roles": ["Support"],
  "created": "2025-01-04T10:30:00Z"
}
```

API 仕様を定義する

API 仕様を定義する

```
{
  "username": "jane.doe",
  "email": "jane.doe@company.com",
  "password": "TempP@ssw0rd!",
  "first_name": "Jane",
  "last_name": "Doe",
  "role": "Support,Finance"
}
```

API 仕様を定義する

POST /auth/roles/{role_id}/users/{user_id}

API 仕様を定義する

API 仕様を定義する

GET /auth/users

API 仕様を定義する

API 仕様を定義する

GET /auth/users/search?search={query}&filters={"role":["Support"]}&page=1&per_page=50

API 仕様を定義する

- API 仕様
- API 仕様
- API 仕様
- API 仕様
- API 仕様

データベース

GET /auth/users/{user_id}

データベース

- ユーザー
- ユーザーグループ
- ユーザーロール
- ユーザーセッション
- ユーザーパスワード

データベース

データベース

データベース

Web UI

1. ユーザー → ユーザー
2. **"ユーザー"*
3. ユーザーグループ
 - ユーザー - ユーザーグループ"Tier2_Support"
 - ユーザー - ユーザーグループ
4. **"ユーザー"*
5. ユーザーグループ

API

POST /auth/roles

admin

データベース

```
{
  "name": "Tier2_Support",
  "description": "Level 2 support team with elevated provisioning access"
}
```

000

```
{
  "id": 45,
  "name": "Tier2_Support",
  "description": "Level 2 support team with elevated provisioning access",
  "permissions": [],
  "users": []
}
```

--	--	--	--	--	--	--

□ □

Web UI

1. $\square\square\square\square\square\square\square \rightarrow \square\square\square\square$
2. $\square\square\square\square\square\square\square\square\square\square\square$
3. $\square\square\square\square\square\square\square\square^{**}\text{"}\square\square\square\square\text{"}^{**}$
4. $\square\square\square\square\square\square\square\square\square\square\square\square$
5. $\square\square^{**}\text{"}\square\square\text{"}^{**} - \square\square\square\square\square\square$

API

POST /auth/roles/{role_id}/permissions

```
{  
  "permission_id": 123  
}
```

□□□□□□□□

```
{  
  "permission_ids": [123, 124, 125]  
}
```

□□□□□“□□□□”□□

□□□□□□□□□□□□□□□□□□

1. □□□□□

```
POST /auth/roles  
{  
  "name": "Provisioning_Specialist",  
  "description": "Can provision services and manage customer  
services"  
}
```

2. □□□□□

```
POST /auth/roles/45/permissions  
{  
  "permission_ids": [  
    1,    # view_customer  
    20,   # view_customer_service  
    21,   # create_customer_service  
    22,   # update_customer_service  
    30,   # view_provision  
    31,   # create_provision  
    40,   # view_inventory  
    50,   # view_product  
  ]  
}
```

□□□□□□□□

□□Web UI□

1. □□□□□□□□
2. □□□□□□□□□□□□□□□□**“X”□“□□”**□□
3. □□□□

□□API□

□□□ DELETE /auth/roles/{role_id}/permissions/{permission_id}

□□□

DELETE /auth/roles/45/permissions/31

□□□□□□□□ create_provision □□□

□□□□□□□□

□□□□□□□□□□

□□Web UI□

1. □□□□□□□□ → □□□□□□
2. □□□□□□□□
3. □□□□□□□□□□
4. □□**“□□”**

API

PUT /auth/roles/{role_id}

```
{
  "name": "Senior_Support",
  "description": "Senior support team with full customer access"
}
```


API

DELETE /auth/roles/{role_id}

##

データベースに「ユーザー」を追加

データベース

Web UI

1. ユーザー → 追加
2. ユーザー
3. ユーザー/パスワード
4. ユーザー**パスワード**

API

データベース

PUT /auth/users/{user_id}

```
{  
  "role": "Support, Finance"  
}
```

データベース

POST /auth/roles/{role_id}/users/{user_id}

□□□□□□

□□□□□□□□

GET /auth/roles/{role_id}/users

□□□□□□□□□□□□□□□□

□□□□□□□□

GET /auth/users/{user_id}

□□□□ roles □□□□□□□□□□□□□□

□□□□□□□□

OmniCRM□□□□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

1. □□□□□□□□“□□□□”
2. □□□□□□□□ - □□□□□□□□□□□□
3. □□□□□□□□ - □□□□□□□□□□□□□□□□□□□□□□□□1□□□□
4. □□□□□□ - □□□□□□□□□□
5. □□□□□□ - □□□□□□□□□□□□□□
 - □□8□□□□
 - □□□□□□□□□□
 - □□□□□□□□□□
 - □□□□□□□□
 - □□□□□□□□□□
6. □□ - □□□□□□□□□□□□□□□□□□□□

API□□□□

1. □□□□□□□□


```
{
  "new_password": "TempP@ssw0rd!",
  "force_change": true
}
```

□□□

- new_password - □□□□□□□□
- force_change□□□□ - □□□true□□□□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□ POST /auth/change_password

□□□

```
{
  "current_password": "OldP@ssw0rd!",
  "new_password": "NewSecureP@ssw0rd!"
}
```

□□□□□□□□□□□□□□□□

□□□□□

- □□□□bcrypt□werkzeug□□□□□□□□□□
- □□□□□□□□
- □□□□□1□□□□□□
- □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□

OmniCRM 2FA

OmniCRM 2FA TOTP QR Code

2FA

Web UI

1. Login →
2.   Please enable 2FA
3. Please enable 2FA**
4. Please
 - TOTP QR Code
 - 10 digit code
5. Please enable 2FA Google Authenticator Authy QR
6. Please enable 2FA
7. Please enable 2FA
8. 2FA

API

1. TOTP

POST /2fa/enable/user/{user_id}

```
{
  "totp_secret": "JBSWY3DPEHPK3PXP",
  "qr_code_url": "otpauth://totp/OmniCRM:user@example.com?secret=JBSWY3DPEHPK3PXP&issuer=OmniCRM",
  "backup_codes": [
    "12345678",
    "23456789",
    "34567890",
    ...
  ]
}
```

2. 验证

POST /2fa/verify-setup/user/{user_id}

```
{
  "code": "123456"
}
```

返回 {"verified": true}

2FA 流程

2FA 流程

1. 用户注册
2. 生成密钥
3. 用户扫描二维码 2FA
4. 用户输入验证码
5. 验证成功
6. 用户登录

🔑 🔒 🔐 🔓

- 2FA 🔑 🔒 🔐 🔓 10 🔑 🔒 🔐 🔓
- 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓
- 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓
- 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓

🔑 2FA 🔑 🔒 🔐 🔓

🔑 🔒 🔐 🔓 POST /2fa/verify/user/{user_id}

```
{  
  "code": "123456"  
}
```

🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓

- **TOTP** 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓 6 🔑 🔒 🔐 🔓
- 🔑 🔒 🔐 🔓 🔑 🔒 🔐 🔓 8 🔑 🔒 🔐 🔓

バックアップコード

バックアップコードを生成する

Web UI

- バックアップコードを生成
- バックアップコードを生成
- バックアップコードを生成
- バックアップコード

API

POST /2fa/backup-codes/regenerate/user/{user_id}

レスポンス

```
{
  "backup_codes": [
    "98765432",
    "87654321",
    "76543210",
    ...
  ]
}
```

2FA

2FAを有効にする

Web UI

- 2FAを有効にする → 2FA
- 2FA
- 2FAを有効にする
- 2FA
- 2FAを有効にする
- 2FA

API

POST /2fa/admin/disable/user/{user_id}

admin

2FA

- TOTP
-
- is_2fa_enabled false

2FA

2FA

POST /2fa/reset-for-new-device/user/{user_id}

```
{
  "backup_code": "12345678"
}
```

TOTP

2FA

- 2FA
-
-
- Google Authenticator, Authy, Microsoft Authenticator
- 2FA

- **2FA** - 2FA

🔒🔒🔒

Web UI

1. →
- 2.
3.
 -
 -
 -
 -
 -
4. **"

API

PUT /auth/users/{user_id}

```
{
  "first_name": "Jane",
  "last_name": "Doe-Smith",
  "email": "jane.doesmith@newcompany.com",
  "phone_number": "+61498765432",
  "role": "Support,Finance"
}
```

- pending_email
-
-

API

APIの仕様を定義する

PUT /auth/users/{user_id}

```
{  
  "deleted": false  
}
```

deletedの値をfalseに設定する

APIの仕様を定義する

API

APIの仕様を定義する

UIの仕様を定義するGDPRの仕様を定義する

API

- API - API
- API - API
- API - API
- API - API
- API - API
- API - API

API

APIの仕様を定義する

- view_* — API
- create_* — API
- update_* — API

- delete_* — 删除/删除

删除/删除**“删除...”**删除/删除

删除/删除

- admin — 删除/删除
- can_impersonate — 删除/删除
- grafana_access — Grafana删除/删除
Grafana删除/删除

删除/删除

- 删除
 - view_customer[] create_customer[] update_customer[] delete_customer
 - 删除/删除
- 删除
 - view_customer_site[] create_customer_site???
update_customer_site[] delete_customer_site
 - 删除/删除
- 删除

- view_customer_contact[] create_customer_contact[]
update_customer_contact[] delete_customer_contact
- [] [] [] [] [] [] [] [] [] []
- [] [] [] [] [] [] [] [] [] [] <administration_attributes>[]
 - view_customer_attribute[] create_customer_attribute[]
update_customer_attribute[] delete_customer_attribute
 - [] [] [] [] [] [] [] [] [] []
- [] [] [] [] [] [] [] [] [] [] <administration_tags>[]
 - view_customer_tag[] create_customer_tag[] update_customer_tag[]
delete_customer_tag
 - [] [] [] [] [] [] [] [] [] []
- [] [] [] []
 - view_customer_service[] create_customer_service[]
update_customer_service[] delete_customer_service
 - [] [] [] [] [] [] [] [] [] []
- [] [] [] []
 - view_customer_activity[] create_customer_activity[]
update_customer_activity[] delete_customer_activity
 - [] [] [] [] [] [] [] [] [] []

[] []

- **Stripe**[]
 - view_customer_stripe_card[] create_customer_stripe_card[]
update_customer_stripe_card[] delete_customer_stripe_card
 - [] [] [] [] [] [] [] [] [] [] Stripe[]
- [] []
 - view_customer_transaction[] create_customer_transaction[]
update_customer_transaction[] delete_customer_transaction
 - [] [] [] [] [] [] [] [] [] []
- [] []
 - view_customer_invoice[] create_customer_invoice[]
update_customer_invoice[] delete_customer_invoice
 - [] [] [] [] [] [] [] [] [] []

□□

- view_communication□ create_communication□ update_communication□ delete_communication
- □□□□ □□□□□□□□

□□□□□

- □□
 - view_inventory□ create_inventory□ update_inventory□ delete_inventory
 - □□□□ □□□□□□□□
- □□□□
 - view_inventory_template□ create_inventory_template□ update_inventory_template□ delete_inventory_template
 - □□□□ □□□□□□□□□□

□□

- view_product□ create_product□ update_product□ delete_product

□□□□**CBC**□

- view_cbc_message□ create_cbc_message□ update_cbc_message□ delete_cbc_message

□□

- □□
 - view_provision□ create_provision□ update_provision□ delete_provision
 - □□□□ □□□□□□□□
- □□□□
 - view_provision_event□ create_provision_event□ update_provision_event□ delete_provision_event

“ ”

GET /auth/permissions

POST /auth/permissions

```
{
  "name": "view_example",
  "description": "Read-only access to example objects"
}
```

GET /auth/roles

POST /auth/roles

```
{
  "name": "Support",
  "description": "Tier-1 support team"
}
```

POST /auth/roles/{role_id}/permissions

```
{
  "permission_id": 123
}
```


DELETE /auth/roles/{role_id}/permissions/{permission_id}

DELETE

DELETE

POST /auth/users

POST

```
{
  "username": "sara",
  "email": "sara@example.com",
  "password": "TempP@ssw0rd!",
  "first_name": "Sara",
  "last_name": "Ng",
  "phone_number": "+61...",
  "role": "Support"
}
```

POST

PUT /auth/users/{user_id}

PUT

```
{
  "role": "Finance"
}
```

PUT

GET /auth/users

GET

- can_impersonate admin

GET

POST /auth/impersonate

请求体

```
{ "user_id": 42 }
```

响应体

POST /auth/stop_impersonation

注意事项

- 需要权限 impersonate
- 返回“成功...” 表示成功 impersonate
- 需要权限 impersonate —— 需要权限
- 需要权限 admin 和 can_impersonate

Grafana 安装

OmniCRM 安装 Grafana 需要安装 grafana_access 和 Grafana 安装

安装 Grafana

安装 grafana_access 和 /grafana 安装 Grafana 安装 OmniCRM 安装 - 安装 Grafana 安装

安装 Grafana 安装

- 安装 grafana_access
- 安装 /grafana
- 安装 Grafana
- 安装 OmniCRM

OmniCRM

Grafana OmniCRM CRM

1. **Grafana** - `/grafana`
2. **ID** - Grafana URL `/grafana/d/abc123/dashboard-name` ID `/d/` `abc123`
3. - ID
4. - OmniCRM “” OmniCRM

- **ID** - Grafana URL
- - OmniCRM

??

Grafana

`grafana_access` Grafana

-
-
-
-

□□□□□□□□ □□□□□□□□□□□□□□□□

“□□□□...”□□□□□□□□□□ □□□□□□/□□□□□□□□□□□□□□□□

□□□□□□□**Grafana**□□□□ □□□□□□□ grafana_access □□□□□□ /grafana □□□□□□
Grafana□□□

顧客サービス

顧客サービスは、顧客のニーズに応えるための重要な役割を果たします。

顧客サービス

- 顧客サービスの重要性
- 顧客サービス <customer_care> の役割 → 顧客 → “顧客サービス”

顧客サービス <customer_care> の役割は、顧客のニーズに応えることです。また、顧客サービス <authentication_flows> の役割は、顧客の認証を行うことです。

顧客サービス

顧客サービスの役割

- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える

顧客サービス

顧客サービスの役割は、顧客のニーズに応えることです。また、顧客サービスは、顧客のニーズに応えるための重要な役割を果たします。

顧客サービス

- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える
- 顧客サービス - 顧客のニーズに応える

- 00 - 0000000000000000

000000

00000000000000000000

000000

0000000

- 000000000000
- 0000
- 00
- 0000

00000

- 00ID
- 000000
- 0000000/000
- 000000

- 密碼

密碼

1. 密碼 → 密碼
2. 密碼 “ ”
3. 密碼
4. 密碼 “ ”
5. 密碼

密碼

- 密碼
- 密碼
- 密碼
- 2FA 密碼

密碼

密碼

□□□□□

□□□□□□

- □□□□ - □□□□□□□□□□“□□ - +44 7700 900123”□
- □□ - □□□□□□□
- □□ - □□□□□□□□□□
- □□□□ - □□□□□□
- □□□□ - □□□□□□□□□□
- □□ - □□□□
- □□□□ - □□/□□□□

□□□□□

- □□□□□□ - □□□□□□□□□□□□□□□□□□□
- □□ - □□□□□□□
- □□□□ - □□□□□□
- □□ - □□□□□□□□□□□□
- □□□□□□ - □□□□□□□□□

□□□□□□□□

- □ □□ - □□□□□□
- □ □□□□ - □ <7 □□□□□□□
- □ □□ - □□□□□□□□□□□□□□□□
- • □□ - □□□□□□

□□□□□

□□□□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□□□□□□□

பொருள்

- பொருள் - பொருள்பெற்று
- பொருள் - பொருள்பெற்று
- பொருள் - பொருள்பெற்று
- பொருள் - பொருள்பெற்று
- பொருள் - பொருள்பெற்று

பொருள்

பொருள்பெற்றுபெற்று + பொருள்பெற்றுபெற்று

- பொருள் **10** - பொருள்பெற்று/பொருள்
- பொருள் **20** - பொருள்பெற்று
- பொருள் **30** - பொருள்பெற்று

பொருள்

பொருள்பெற்றுபெற்று

பொருள்

- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்

பொருள்

- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்

பொருள்

பொருள்

பொருள்

- 0000 - 0000
- 0000 - 000000
- 00**MMS** - MMS000000
- 0000 - 0000000000

0000

0000000000000000

0000

- 0000 - 0000
- 0000 - 0000000000
- 0000 - 0000000000
- 0000 - 00/0000

0000

000000000000000000

00

0000000000000000

□□□□□□

- □□□□ - □□□□□
- □□ - □□□□□□
- □□□□ - □□□□□□
- □□ - □□□□□
- □□ - □□□□□□□□□□
- □□ - □□PDF□□□□□

□□□□□

1. □□□ □□ → □□
2. □□□□□□□□□□□□
3. □□ “□□□□” □□
4. □□□□□□□□□□□□□□
5. □□□□
6. □□□□□□□□

□□□□□

1. □□□□□□ “□□” □□

2. PDF 生成

3. 保存

PDF

生成

保存

- 1 - 1
- 2 - 2
- 3 - 3
- 4 - 4
- 5 - 5

PDF

生成

保存

- 1 - 1
- 2 - 2
- 3 - 3
- 4 - 4

PDF <payment_methods> 生成

PDF / PDF

生成

保存

1. PDF 1 → PDF 2 → “PDF”
2. 生成

3. 套餐選擇
4. 套餐內容
5. 套餐價格
6. 套餐特點
7. 套餐詳情

套餐詳情

- 套餐 - 1GB、5GB、10GB、20GB、50GB
- 套餐 - 套餐內容
- 套餐 - 套餐特點
- 套餐 - 套餐價格£5、£10、£20、£50、£100

套餐 <features_topup_recharge> 套餐詳情

套餐詳情

套餐詳情

套餐詳情

- 套餐 - 套餐內容
- 套餐 **IP** - 套餐內容IP
- 套餐 - IPTV 套餐內容
- 套餐 - 套餐價格
- 套餐 - 套餐特點

套餐詳情

1. 套餐 1 → 套餐 2 → “套餐 3”
2. 套餐詳情
3. 套餐詳情 “套餐 4”
4. 套餐詳情 + 套餐 5
5. 套餐 6
6. 套餐詳情

□□□ □□□□ - □□□□ <csa_modify> □□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□

- □□□□□□□□□□
- □□□□□□□
- □□□□□□□7□□3□□1□□
- □□□□□□□□□□□□□□□□
- □□□□□/□□
- □□□□□□□
- □□□□□□□□□□□□□□

□□□□□□□

- □□□□□□□
- □□□□□□□□
- □□□□□□□□□10%□
- □□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□

□□□□□□

□□□□□□□□

- □□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□<20%□□□□□
- □□□□□□□□□□□□□□□□

環境問題

環境問題の重要性

環境問題の現状

- 環境問題の重要性 ↔ 環境
- 環境問題の現状
- 環境問題の重要性
- 環境問題
- 環境問題
- 環境問題
- 環境問題

環境問題

- 環境問題の重要性
- 環境問題の現状
- 環境問題
- 環境問題
- 環境問題/環境

環境問題

環境問題の重要性

環境問題

1. 環境問題 → 環境
2. 環境問題
3. 環境問題
4. 環境 “環境問題”
5. 環境問題
6. 環境問題
7. 環境問題

□□□

- □□□□ - □□□□□□□□□□
- □□ - □□□□□□□□□□□□□□
- □□ - □□□□□□□
- □□ - □□□□□□□□□□□□□□□□□□□□

□□□□□□□

- □□ `can_impersonate` □ `admin` □□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□
- □□□□□□□□□□□□□

□□□ □□□□ - □□□□ `<customer_care>` □□□□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□

- 網路連線
- 網路通訊
- 網路通訊設備
- 網路通訊設備
- 網路eSIM網路通訊設備
- 網路通訊
- GPS網路通訊設備

網路Web網路 (PWA)

- 網路通訊設備
- 網路通訊設備
- 網路通訊設備
- 網路通訊設備

網路通訊設備

網路通訊設備網路通訊設備

網路通訊設備

1. 網路通訊設備 “網路通訊”
2. 網路通訊設備
3. 網路通訊設備網路通訊設備1網路
4. 網路通訊設備
5. 網路通訊設備網路通訊設備
6. 網路通訊設備

網路通訊

- 網路8網路
- 網路通訊設備
- 網路通訊設備
- 網路通訊
- 網路通訊設備 (!@#\$%^&*)

00000

50000000

- 00003000
- 000000000000
- 000000000000

0000000

00000

1. 002FA000000
2. 00000000000
3. 000000000000000
4. 000000000000000
5. 000000000000000
6. 00000000000000002FA
7. 0000/000000000

00000

1. 000000000010-20%
2. 000000000000000
3. 000000000000000
4. 000000000000000
5. 0000000000000000000

00000

000000000000000

1. 00000/00000000000
2. 000000000000000
3. 0000000000000000000
4. 0000ID0000000000000

API

API

API

1. → **API**
2. “**API**”
- 3.
- 4.
5. API

API

-
-
-
-
-

API <concepts_api>

API 1.0.0

API 1.0.0

API 1.0.0

API 1.0.0

API 1.0.0 **2FA** API 1.0.0

API 1.0.0 2FA API 1.0.0 2FA API 1.0.0

API 1.0.0

API 1.0.0

API 1.0.0

API 1.0.0 API 1.0.0

API 1.0.0

- `<customer_care>` - API 1.0.0
- `<payment_methods>` - API 1.0.0
- `<features_topup_recharge>` - API 1.0.0
- `<csa_service_usage>` - API 1.0.0
- `<billing_overview>` - API 1.0.0
- `<authentication_flows>` - API 1.0.0
- `2FA <2fa>` - API 1.0.0
- `<csa_modify>` - API 1.0.0

OmniCRM 概要

OmniCRM は Omnitouch を活用した BSS/OSS 向け CRM ソリューションです。
顧客との関係性を強化し、業務効率を向上させるためのツールです。

概要

特徴

主な特徴

1. 顧客情報の一元管理
2. 顧客とのコミュニケーション履歴の記録
3. 顧客のニーズに応じた提案の作成
4. 顧客の満足度の向上
5. 業務効率の向上

構成

- 顧客情報管理システム
- 顧客とのコミュニケーション履歴管理システム
- 顧客のニーズに応じた提案作成システム
- 顧客の満足度向上システム

導入メリット

導入効果

1. 顧客情報の一元管理
2. 顧客とのコミュニケーション履歴の記録
3. 顧客のニーズに応じた提案の作成
4. 顧客の満足度の向上
5. 業務効率の向上

🔒 🔒 🔒 🔒

- 📱 - 🔒
- 📧 - 🔒 🔒 🔒 🔒 🔒 🔒 🔒
- **API** 🔒 - 🔒 🔒 🔒 🔒
- 📱 - 🔒 2FA 🔒 🔒

🔒 🔒 🔒 🔒

🔒 🔒 🔒 🔒 🔒 🔒 🔒 🔒

- 📱 🔒 🔒 🔒 🔒 - 🔒 🔒 🔒 🔒
 - 📧 🔒 🔒 - 🔒 🔒 🔒 🔒 🔒 🔒
 - 📱 🔒 🔒 🔒 🔒 - 🔒 🔒 🔒 🔒 🔒 🔒
 - 📱 🔒 🔒 - 🔒 🔒 🔒
 - 📱 🔒 🔒 - 🔒 🔒 🔒 🔒 🔒
-

目录

章节...	页码
第一章	1
第二章	2
第三章	3
第四章	4
第五章	5
第六章	6
第七章	7
第八章	8
第九章	9
第十章	10
第十一章	11
第十二章	12
第十三章	13
第十四章	14
第十五章	15
第十六章	16
第十七章	17
第十八章	18
第十九章	19
第二十章	20
第二十一章	21
第二十二章	22
第二十三章	23
第二十四章	24
第二十五章	25
第二十六章	26
第二十七章	27
第二十八章	28
第二十九章	29
第三十章	30
第三十一章	31
第三十二章	32
第三十三章	33
第三十四章	34
第三十五章	35
第三十六章	36
第三十七章	37
第三十八章	38
第三十九章	39
第四十章	40
第四十一章	41
第四十二章	42
第四十三章	43
第四十四章	44
第四十五章	45
第四十六章	46
第四十七章	47
第四十八章	48
第四十九章	49
第五十章	50
第五十一章	51
第五十二章	52
第五十三章	53
第五十四章	54
第五十五章	55
第五十六章	56
第五十七章	57
第五十八章	58
第五十九章	59
第六十章	60
第六十一章	61
第六十二章	62
第六十三章	63
第六十四章	64
第六十五章	65
第六十六章	66
第六十七章	67
第六十八章	68
第六十九章	69
第七十章	70
第七十一章	71
第七十二章	72
第七十三章	73
第七十四章	74
第七十五章	75
第七十六章	76
第七十七章	77
第七十八章	78
第七十九章	79
第八十章	80
第八十一章	81
第八十二章	82
第八十三章	83
第八十四章	84
第八十五章	85
第八十六章	86
第八十七章	87
第八十八章	88
第八十九章	89
第九十章	90
第九十一章	91
第九十二章	92
第九十三章	93
第九十四章	94
第九十五章	95
第九十六章	96
第九十七章	97
第九十八章	98
第九十九章	99
第一百章	100

📁📁📁

📁📁📁📁

- 📁📁📁 - 📁📁📁📁📁📁📁
- 📁📁📁📁📁 - 📁📁📁📁📁
- **Ansible** 📁📁📁 - 📁📁📁📁📁

📁📁📁📁

- 📁📁📁📁📁📁 - 📁📁📁📁📁
- 📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁📁📁📁
- 📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁📁

📁📁📁📁

- 📁📁📁 - 📁
- 📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁
- 📁📁📁 - 📁📁📁
- 📁📁📁📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁

📁📁📁📁📁

- 📁📁📁 - 📁📁📁📁📁
- 📁📁📁 - 📁📁📁📁
- 📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁📁📁📁

API

- [OmniCRM](#) - [API](#)
- [API](#) - [API](#)
- [API](#) - [API](#)
- [API](#) - [API](#)

API

- [API](#) - [API](#)
- [API](#) - [2FA](#)
- [API](#) - [API](#)

API

- [Stripe](#) - [API](#)
- [Mailjet](#) - [API](#)
- [API](#) - [REST API](#)

API

- [API](#) - [API](#)
- [API](#) - [API](#)
- [API](#) - [API](#)

API

- [API](#) - [API](#)
- [API](#) - [API](#)

OmniCRM API

OmniCRM API

📁 目錄

📄 01 - OmniCRM 使用 **Ansible** 部署及配置 SIM 卡管理系統

📄 02

- 安裝 Ansible
- 配置 Ansible 環境
- 部署 SIM 卡管理系統
- 配置 SIM 卡管理系統
- 測試 SIM 卡管理系統

📄 03 | **Ansible** 部署

📁 目錄

📄 **CGRateS** 部署及配置 **Stripe** 支付系統

📄 04

- 安裝 CGRateS
- 配置 CGRateS
- 部署 Stripe 支付系統
- 配置 Stripe 支付系統
- 測試 Stripe 支付系統
- 部署 Stripe 支付系統

📄 05 | 部署 | 測試

📁 目錄

📄 06 - 部署及配置 SIM 卡管理系統

📄 07

- 安裝 Ansible
- 配置 Ansible 環境

- 000000000000
- 00000000
- 000000000000
- Google 000000000000

000000 | 0000 | 0000

0000

0000000000000000 - 0 SIM 00000000 IP 0000

000000

- 00 SIM 00000000
- 00000000 IP 0000
- 000000000000
- 000000000000
- 00000000
- 00000000

000000

0000000000

00 000000 0000000000000000

0000000000

- 000000000000
- 000000000000
- 00000000
- 00 PDF 000000
- 00000000
- 000000000000

0000000000 | 00000000

API

API 是系統與系統之間溝通的橋樑

API 類型

- **REST** API
- **GraphQL** (2FA) API TOTP API
- **SOAP** (RBAC) API
- API 代理
- API 聚合
- API 門戶

API 安全 | **RBAC** | **2FA**

API 安全

OmniCRM 安全策略

- **CGRateS** - 安全策略
- **Stripe** - 安全策略
- **Mailjet** - 安全策略
- **Google** - 安全策略
- **RESTful API** - 安全策略

API | **Stripe** | **Mailjet**

API 安全

API 安全 - 安全策略

API 安全

- API - 安全策略
- API - 安全策略 + API + API
- API - 安全策略
- API - 安全策略

OmniCRM 2024 年 10 月 10 日

OmniCRM | 2024 年 10 月 10 日

OmniCRM

OmniCRM 2024 年 10 月 10 日

OmniCRM

- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日

OmniCRM

OmniCRM ? ?

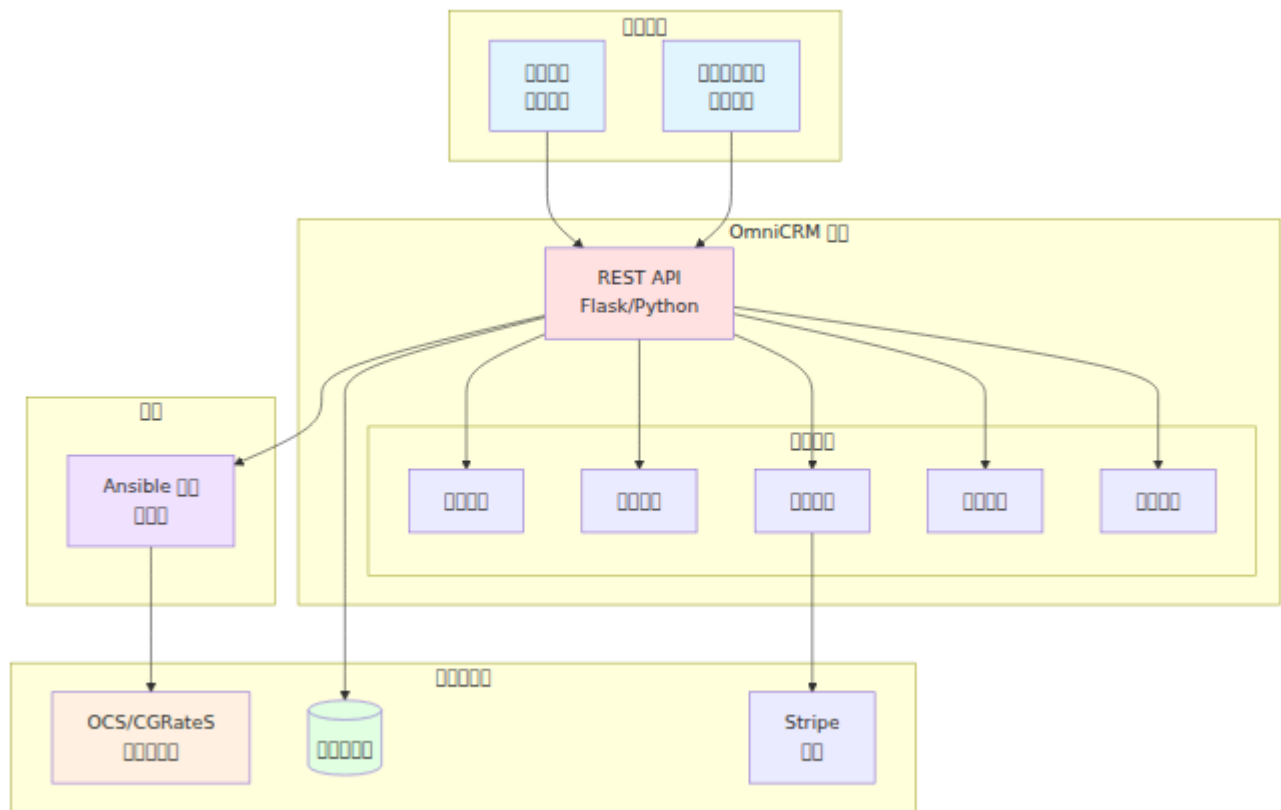
OmniCRM 2024 年 10 月 10 日

OmniCRM

- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日
- OmniCRM 2024 年 10 月 10 日

OmniCRM | 2024 年 10 月 10 日

概要



構成

概要

構成

OmniCRM 構成

- = 構成
- = 構成

Ansible 構成

構成

📁 目錄

📁 Ansible 配置管理工具使用說明書

📁 目錄

📁 配置管理工具使用說明書 - 配置管理工具 OmniCRM

📁 目錄

📁 配置管理工具使用說明書 CGRateS 配置管理工具使用說明書

📁 目錄

JWT 認證 2FA RBAC 配置管理工具使用說明書

📁 API

📁 REST API 配置管理工具 OmniCRM 配置管理工具使用說明書

📁 目錄

📁 目錄

- 📁 目錄 配置管理工具
- 📁 目錄 配置管理工具
- 📁 目錄 配置管理工具
- 📁 目錄 配置管理工具

📁 目錄

📁 配置管理工具使用說明書 目錄 配置管理工具

OmniCRM - 株式会社 BSS/OSS

2025-12-23