

OmniCRM 概要

OmniCRM は Omnitouch を活用して、BSS/OSS 業務を効率化するための
プラットフォーム - 統合顧客関係管理

概要

目的

目的

1. 効率化 - 業務効率化
2. 透明化 - 業務透明化
3. 連携 - 部門間連携
4. 自動化 - 業務自動化
5. 最適化 - 業務最適化

機能

- 顧客管理 - 顧客情報管理
- 問い合わせ管理 - 問い合わせ履歴管理
- 履歴管理 - 履歴情報管理 (CDR)
- レポート - 業務レポート
- 連携 - 外部システム連携

構成

構成

1. 顧客管理 - 顧客情報
2. 問い合わせ管理 - 問い合わせ
3. 履歴管理 - 履歴情報

4. 帳號 - 密碼

5. 帳號 - 密碼

帳號

- 帳號 - 密碼
- API 密鑰 - 密碼
- 帳號 - 密碼 2FA 密碼

密碼

密碼

- 密碼 - 密碼
 - 密碼 - 密碼
 - 密碼 - 密碼
 - 密碼 - 密碼
 - 密碼 - 密碼
-

目录

目录...	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** □□□□ - □□□□□□□□

□□□□

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

□□

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

□□□□

- □□ - □□□□□□□□□□

□□□□

- □□□□ - □□
- □□□□ - □□□□□□□□
- □□□□ - □□□□□
- □□□□ - □□□□□
- □□□□□□□□□□ - □□□□□□□□
- □□□□□□□ (CDR) - □□□□□□□□□□□□□□
- □□□□□□ - □□□□□□

☐☐☐☐☐

- ☐☐☐☐ - ☐☐☐☐☐☐
- ☐☐☐☐☐☐ - ☐☐☐☐☐☐
- ☐☐☐☐ - ☐☐☐☐☐☐
- ☐☐ - ☐☐☐☐☐☐
- ☐☐☐☐ - ☐☐☐☐☐☐
- ☐☐☐☐ - ☐☐☐☐☐☐☐☐

☐☐

- ☐☐☐☐ - ☐☐ OmniCRM
- ☐☐☐☐ - ☐☐☐☐
- SIM ☐☐☐ - SIM ☐☐☐☐☐☐
- eSIM ☐☐☐☐ - eSIM ☐☐☐☐☐
- API ☐☐ - API ☐☐☐☐
- ☐☐☐☐☐ - ☐☐☐☐☐☐☐☐☐☐

☐☐☐☐☐☐

- ☐☐☐☐☐☐☐ - ☐☐☐☐☐☐☐☐☐☐
- ☐☐☐☐☐ - ☐☐ 2FA
- ☐☐☐☐☐☐☐☐☐☐ - ☐☐☐☐☐☐☐☐☐

☐☐

- ☐☐☐☐☐☐☐☐☐ (EDM) - ☐☐☐☐☐☐☐
- Mailjet ☐☐ - ☐☐☐☐☐☐☐
- ☐☐ API - ☐☐☐☐☐☐☐
- API ☐☐ - REST API ☐☐

☐☐☐☐☐

- ☐☐☐☐☐☐☐ - ☐☐☐☐☐☐☐
- ☐☐☐☐☐ - ☐☐☐☐☐☐☐

- [Ansible 入門](#) - [インストール](#)

📄

- [Ansible 入門](#) - [インストール](#)
- [Ansible 入門](#) - [インストール](#)

OmniCRM 構築ガイド

OmniCRM 構築ガイド

📄

OmniCRM 構築ガイド **Ansible** 入門 [Ansible 入門](#) [Ansible 入門](#) SIM [Ansible 入門](#) [Ansible 入門](#)

📄

- [Ansible 入門](#)
- [Ansible 入門](#)
- [Ansible 入門](#)
- [Ansible 入門](#)
- [Ansible 入門](#)

[Ansible 入門](#) | [Ansible 入門](#)

📄

[CGRateS](#) [CGRateS](#) Stripe [Stripe](#)

📄

- [CGRateS](#)
- [Stripe](#)
- [Stripe](#)

- 
- 
- 

□□□□□□ | □□□□ | □□□□

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

□ □

5/5

- 00000000000000
- 0000000000000
- 0000000000000
- 00000000
- 00000000000000
- Google Maps 000000000000

□□□□□□ | □□□□ | □□□□

□ □ □ □

□□□□□□□□□□□□□□ - □ SIM □□□□□□□ IP □□□□

□□□□□

- SIM 卡
- IP 地址
- 手机号码
- 电子邮箱
- 姓名
- 性别

Page 10 of 10

API 端點

API 端點清單

API 端點清單

- API 端點清單
- API 端點清單
- API 端點清單
- API 端點 PDF
- API 端點清單
- API 端點清單

API 端點清單 | API 端點清單

API 端點

API 端點清單

API 端點

- API 端點 **JWT** 端點
- API 端點 **(2FA)** 端點 TOTP 端點
- API 端點 **(RBAC)** 端點
- API 端點
- API 端點
- API 端點

API 端點 | **RBAC** 端點 | **2FA** 端點

API 端點

OmniCRM 端點

- **CGRateS** - 端點
- **Stripe** - 端點
- **Mailjet** - 端點

- **Google Maps** - 地圖服務
- **RESTful API** - 資料服務

API 介紹 | Mailjet 介紹

API 介紹

API 介紹 - 資料服務

API 介紹

- API - 資料服務
- API - 資料服務 + API + API
- API - 資料服務
- API - 資料服務

API 介紹 - 資料服務

API 介紹 | 資料服務

API 介紹

API 介紹 - OmniCRM API 資料服務

API 介紹

- API 介紹
- API 介紹
- API 介紹
- API 介紹

API 介紹

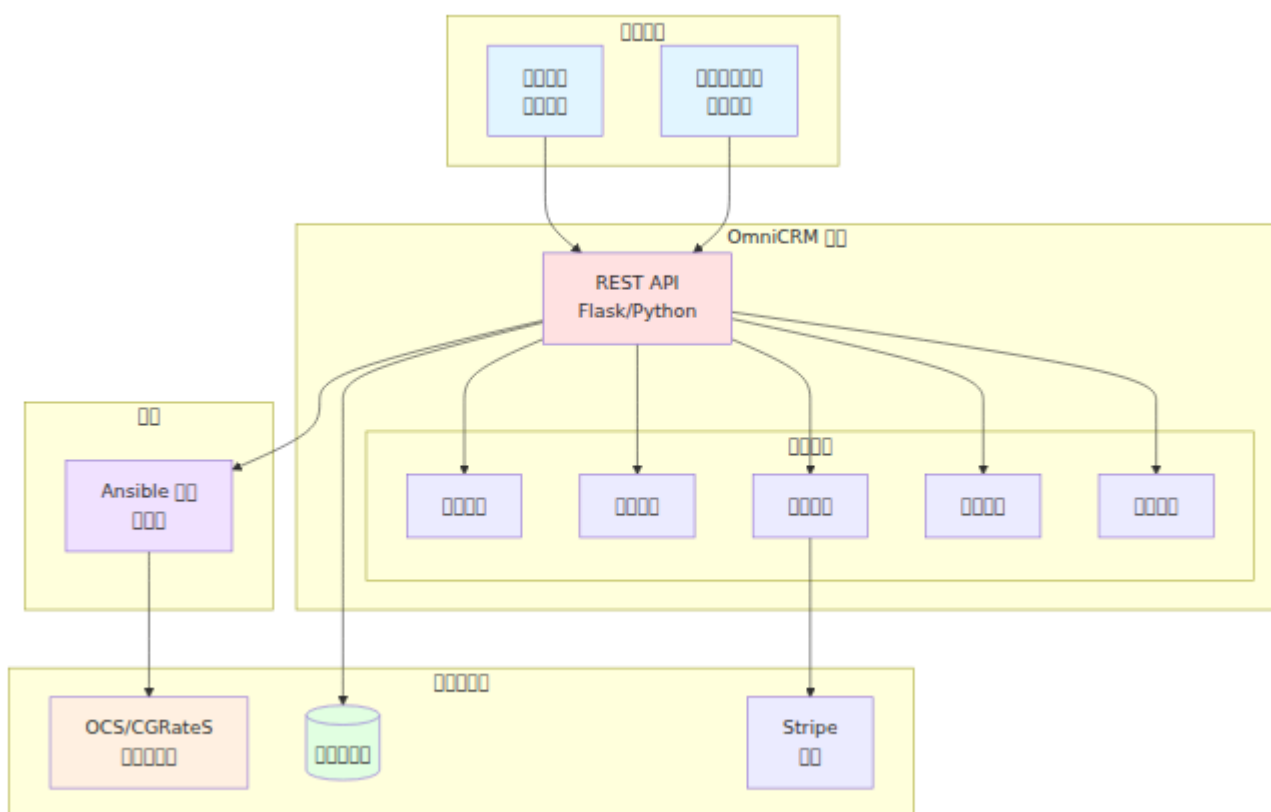
API 介紹

API 介紹 - 資料服務

API 介紹

- | | | | | | | | | |
|---|---|---|---|--|---|---|---|---|
| □ | □ | □ | □ | | □ | □ | □ | □ |
|---|---|---|---|--|---|---|---|---|

--



Page 1 of 1

1111

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

OmniCRM

- $\text{IP} = \text{IP address of the client} \text{ "IP address"}$
- $\text{IP} = \text{IP address of the server} \text{ "IP address"}$

Ansible

□ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □

OmniCRM ██████████



`CGRateS`

Ansible | 安装

□ □ □ □



Stripe

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

□ □ □ □

```

00 (ABC 00)
├ 000 (00.0000 - 00, 0.0 - 00)
├ 00 (00, 000)
└ 00
    ├── 000 - 00
    ├── 0000 - 00.000
    └ VoIP - 000

```

□ □

Page 10 of 10

OmniCRM

--	--	--	--	--

□□□□□□□□□□□□□□ - □□ juggling □□□□□□□□□□

--	--	--	--	--

Ansible

□ □ □ □ □

████████████████████ - ████████████████████ OmniCRM

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

CGRateS

□ □ □ □

JWT OAuth2FA RBAC

API

REST API OmniCRM

□ □ □ □

- 
- 
- 
- 

□□□□□□

□□□□□□□□□□ □□□ □□□□□□

OmniCRM - □□□□□□□□ BSS/OSS

□□□□2025-12-23

雙重認證 (2FA)

雙重認證是什麼？

OmniCRM 會要求您 (2FA) 透過時間一次性密碼 (TOTP) 進行雙重認證，以確保您的帳戶安全。

透過 RBAC 設定 2FA 可確保您的帳戶安全。

如何設定

2FA 設定

1. 前往系統管理頁面。
2. 點選「系統設定」。
3. 點選「安全設定」。
4. 點選「Google Authenticator」或「Authy」或「Microsoft Authenticator」，並按照指示進行設定。

2FA 優點

2FA 優點

1. 防止未經授權的訪問。
2. 防止密碼洩露。
3. 防止釣魚攻擊。
4. 防止惡意軟體攻擊。

🔒 2FA

📋 設定

🔒 2FA 設定

1. 🔒 2FA を有効にする
2. 🔒 2FA の種類を選択
3. 🔒 2FA のコードを入力
4. 🔒 2FA のコードを確認
5. 🔒 2FA のコードを保存

📋 設定

🔒 2FA

🔒 2FA を有効にするには、以下の手順に従ってください。

- 🔒 2FA の種類を選択
- 🔒 2FA のコードを入力

- 2FA 2FA

2FA

2FA 2FA `totp_secret` 2FA 2FA

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 Backup Codes
- API Endpoints for 2FA Backup Codes
- API Endpoints for 2FA Backup Codes

API Endpoints

API Endpoints for TOTP (Time-based One-time Password) using Google Authenticator, Authy, Microsoft Authenticator, 1Password, etc.

API Endpoints for 2FA (Two-Factor Authentication)

API 概念

API 是应用程序编程接口，是用于与 OmniCRM API 交互的 API 接口



API 概念 concepts_api

API

API 是应用程序编程接口，是用于与 OmniCRM API 交互的 API 接口

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

JWT 令牌/令牌API 令牌

API 接口

接口

接口

接口 `MANAGE_API_KEYS` 接口

API 接口

接口 API 接口

接口

- **API** - API 的 名称 和 版本 信息 “API 名称” “API 版本”
- **API 文档** - API 的 文档 信息
- **API 接口** - API 的 接口 信息
- **API 参数** - API 的 参数 信息
- **API 返回** - API 的 返回 信息
- **API 错误** - API 的 错误 信息
- **API 安全** - API 的 安全 信息

API 文档

API 接口

API 参数

- **API 名称** - API 的 名称 信息
- **API 版本** - API 的 版本 信息
- **API 接口** - API 的 接口 信息 30 分钟内 有效
- **API 参数** - API 的 参数 信息

API 文档

1. API 名称

API 名称 和 版本 信息 + API 版本

2. API 接口

API 接口 信息

API 名称 和 版本 信息 “API 名称” “API 版本”

API 名称 和 版本 信息 “API 名称” “API 版本”

API 名称 和 版本 信息 “API 名称” “API 版本”

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒ ☐

[] []

-
- “”“”“”
-

-
- “ 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

-
-
-
-
-
-

- 認證服務
- 認證服務
- 認證服務服務服務
- 認證服務服務服務服務

API 認證服務

API 認證服務

1. 認證服務 : 認證服務
2. 認證 “認證”
3. 認證服務服務服務服務
4. 認證 “認證”

認證服務

- 認證
- 認證
- 認證服務
- 認證

認證服務

- 認證服務服務服務服務服務
- 認證服務
- 認證服務

API 認證服務

API 認證服務

認證

- 認證服務服務
- 認證服務服務服務服務
- 認證服務

- 00000000

0000

- 0000000000 30 0000
- 000/000000
- 0000000000

00

- 000000
- 00000000
- 0000000
- 000000000

000

- 0000/00
- 0000
- 0000000000

000000

API 00000000

000

0000000000000000

000000

0000000000

- 0000
- 000
- 0000000000 30 00
- 00

API

API

- API
- API
- API
- API

API

API

- API 32
- API
- API sk_live_

API

CRM

- CRM
- CRM
- CRM
- CRM

API

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

API

- API - API
- API
- API/API
- API

API

- API API
- API
- API“API”
- API

API

API

1. API
2. API
3. API
4. API

API

API “401 Unauthorized”

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

API “API API”

- 所有 数据表数据表数据表
- 数据表
 - 所有 API 数据表数据表
 - 数据表数据表
 - 数据表数据表

API 数据表

- 所有 数据表数据表30 数据表
- 数据表
 - 数据表数据表数据表
 - 数据表数据表数据表
 - 数据表数据表

数据表 API 数据表

- 所有 数据表数据表
- 数据表
 - 数据表数据表
 - 数据表数据表/数据表
 - 数据表数据表数据表

API 数据表数据表数据表

API 数据表数据表数据表 API 数据表数据表数据表

数据表 API 数据表

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

数据表 API 数据表

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. 概念 API 概念

概念

- `concepts_api` - API 概念
- `rbac` - 权限概念
- `2fa` - 二次验证概念

API

APIは、異なるシステム間でデータをやり取りするための標準的な方法です。

APIの種類には、REST API、SOAP API、GraphQL APIなどがあります。

APIは、Webサービス、モバイルアプリ、IoTデバイスなど、さまざまなシステムと接続するために使用されます。

APIの種類

APIの種類は、その通信方式によって異なります。

1. REST API

REST APIは、HTTPプロトコルを使用するAPIです。

- `external_crm_id` = "SF-12345" (SalesforceのID)
- `legacy_system_id` = "OLD-CRM-789" (旧システム)
- `hubspot_contact_id` = "12345678" (HubSpotのID)

2. SOAP API

SOAP APIは、XMLを使用してデータをやり取りするAPIです。

- `billing_method` = "quarterly" (四半期ごとの請求)
- `auto_provision` = "true" (自動プロビジョニング)
- `support_tier` = "premium" (プレミアムサポート)
- `credit_limit` = "10000" (クレジット制限)

3. GraphQL API

GraphQL APIは、クライアントが必要なデータのみを取得できるAPIです。

- `gdpr_consent_date` = "2025-01-01" (GDPR同意日)
- `tax_exempt` = "true" (課税免除)
- `regulatory_entity` = "FCC-123456" (規制機関)

4. 設定

設定項目

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

5. 設定

設定項目

- `radius_username_format` = "email" (必須) RADIUS (必須)
- `vlan_id` = "100" (必須)
- `ipv6_enabled` = "true" (必須)

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
- 設定項目

UI 仕様書

概要

目的

- システム概要
- 利用対象者
- 機能概要
 - 機能A
 - 機能B
 - 機能C
 - 機能D

詳細仕様

機能A

- 機能A概要
- 利用対象者
- 機能A詳細
- 入力項目
 - 外部システムID/外部システムID external_crm_id
 - 外部システムID SF-12345
- 出力項目

機能B

- 外部システムID external_system_id ✓
- 外部システムID external system id ✗
- 機能B詳細
- 機能B詳細

その他

備考

1. □□□□□□□□
2. □□ □□ □□
3. □□□□□□□□□□
4. □□ □□□□□□
5. □□□□□□□□
6. □□ □□□□

[illegible]

□ □ □ □

□□□□□□

1.
2.
3.
4.
5.

[illegible]

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

API

□□□□API□□□□□□□□□□

□ □ □ □ □ □ □

```
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"
}
```

PUT update_attribute

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"
}
```

Python

Example

Python code to delete a customer attribute

```
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']}")
```

❗ 注意

API 実装のサンプルコードは、実際の環境で動作しない可能性があります。

API 仕様

1. 入力

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

2. 出力

- 顧客 ID (150 文字以内)
- 課金方法 ("true"/"false")
- ISO 8601 形式の時刻 ("2025-01-04 10:30:00")

- 使用JSON格式返回数据

3. 接口

- 获取用户信息
- 获取用户列表
- 获取用户详情

4. 接口

- 获取用户信息
- 获取用户列表/获取用户详情
- 获取用户详情

5. 接口

- 获取用户信息
- 获取用户列表
- 获取用户详情 `api_key_v2`

接口

接口

接口

```
# 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

```
# 更新属性
PUT /crm/attribute/
{
  "customer_id": 123,
  "attribute_name": "billing_cycle",
  "attribute_value": "quarterly"
}

# 验证属性
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
)
```

验证属性

验证属性

```
# 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` form
- `update_customer_attribute` form

UI components

- `update_customer_attribute` form
- `attribute_id` field
- `attribute_value` field

UI components

- `attribute_id` field

-
- 

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

- 150
-
-

OmniCRM

OmniCRM は、CRM 機能を提供するためのオープンソースのソフトウェアです。

インストール

OmniCRM をインストールするには、以下の手順に従ってください。

前提条件

- `OmniCRM-API/crm_config.yaml` を取得
- `YAML` を編集
- `API` を起動
- `UI` を起動

インストール手順

- `crm_config.yaml` → `ui`: `GET /crm/public/ui-config` を取得
- `JSON` から `YAML` に変換
- `API` を起動 / `UI` を起動
- `Grafana` を起動 / `Web` を起動

環境変数設定

- `OmniCRM-UI/.env` を `.env.example` からコピー
- `VITE_*` を設定
- `UI` を起動
- `/crm/public/ui-config` から `runtimeConfig.js` を取得

`omnicrm-ui` Docker を起動して `UI` を起動 (`VITE_*` を設定)

crm_config.yaml (crm_config.yaml)

`crm_config.yaml` の設定項目


```
[[[ /OmniCRM/OmniCRM-API/crm_config.yaml
```

55555

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

000

- `username` - MySQL ユーザー名
- `password` - MySQL パスワード
- `server` - MySQL サーバ IP 指定 (localhost)

□ □ □ □ □ □

- `omnicrm`
- `3306`MySQL
- `mysql+pymysql://username:password@server/omnicrm`

[illegible]

□□□ □□□□□□ .env □□□□□□□□

□ □ □ □

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

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

Configuration

- `hss`
- `hss_peers`
- `apn_list`
- `apn_id`

Configuration

HSS Configuration

Configure HSS  configuration

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

Configuration

- `hss_peers` - HSS peers
- `apn_list` - APN ID list

Configuration

Mailjet Configuration

OmniCRM  Mailjet configuration

```
mailjet:
  api_key: your_mailjet_api_key
  api_secret: your_mailjet_api_secret

# 顧客からの問い合わせ
api_crmCommunicationCustomerWelcome:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977509
  subject: "お問い合わせ YourCompany"

# 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 - "

# 請求書の催促
api_crmCommunicationCustomerInvoiceReminder:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 5977570
  subject: "請求書の催促"

# 管理者からのユーザー登録完了メール
api_crmCommunicationUserWelcome:
  from_email: "admin@yourcompany.com"
  from_name: "Your Company Admin"
  template_id: 6118112
  subject: "ようこそ"

# ユーザーからのパスワードリセット
api_crmCommunicationUserPasswordReset:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 6735666
  subject: "パスワードリセット"

# ユーザーからのパスワードリセット成功
api_crmCommunicationUserPasswordResetSuccess:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
```

```

template_id: 6118378
subject: "🔒🔒🔒🔒"

# 📧📧📧📧
api_crmCommunicationUserPasswordChange:
  from_email: "security@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 6118423
  subject: "🔒🔒🔒🔒"

# 📧📧📧
api_crmCommunicationEmailVerification:
  from_email: "verify@yourcompany.com"
  from_name: "Your Company Verification"
  template_id: 6267350
  subject: "🔒🔒🔒🔒🔒🔒🔒🔒"

# 📧📧📧📧
api_crmCommunicationsBalanceExpired:
  from_email: "alerts@yourcompany.com"
  from_name: "Your Company Alerts"
  template_id: 7238252
  subject: "🔒🔒🔒🔒🔒"

# 📧📧📧📧
api_crmCommunicationsBalanceLow:
  from_email: "alerts@yourcompany.com"
  from_name: "Your Company Alerts"
  template_id: 7238263
  subject: "🔒🔒🔒🔒🔒"

# eSIM 📧📧📧
# 📧 template_id 📧📧📧📧 Mailjet 📧📧📧📧📧📧📧
# 📧📧📧 0 📧📧📧📧📧 HTML 📧📧
api_crmCommunicationEsimDelivery:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 0
  subject: "📧📧 eSIM 📧📧📧"

```

eSIM 📧📧📧

📧 eSIM 📧📧📧📧📧📧📧📧

```
api_crmCommunicationEsimDelivery:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 0
  subject: "📶 eSIM 📄📄📄"
```

📄📄📄📄

- 📄📄📄📄📄 eSIM
- 📄 eSIM 📄📄📄📄📄

📄📄📄📄📄 **Mailjet** 📄📄📄📄

- `{{ var:customer_name }}` - 📄📄📄📄
- `{{ var:iccid }}` - eSIM ICCID
- `{{ var:activation_code }}` - LPA 📄📄📄📄📄
- `{{ var:has_qr_code }}` - 📄📄📄📄📄 QR 📄📄📄📄📄 true
- `{{ var:has_activation_code }}` - 📄📄📄📄📄📄📄📄📄 true

📄📄📄📄

📄 `template_id` 📄📄 `0` 📄📄OmniCRM 📄📄📄 eSIM QR 📄📄📄📄📄📄 HTML 📄📄📄📄📄📄
`branding` 📄📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄

📄 OmniCRM 📄📄📄 HTML 📄📄📄📄 Mailjet 📄📄📄📄📄📄📄 `branding` 📄📄📄📄📄📄

```
branding:
  brand_name: 'YourCompany'
  primary_dark: '#594712'
  dark_color: '#32280B'
  accent_bg: '#f5f0e0'
```

📄📄📄

項目	説明	型	例
brand_name	ブランド名	文字列 from_name	'ブランド名'
primary_dark	プライマリダーク	整数 #594712	'#1a73e8'
dark_color	ダークカラー	整数 #32280B	'#0d47a1'
accent_bg	アクセント背景	整数 #f5f0e0	'#e3f2fd'

注意事項

- eSIM 送信時に `api_crmCommunicationEsimDelivery.template_id` に 0 を指定

Mailjet 設定

- Mailjet に登録 (<<https://app.mailjet.com>>)
- API キー → 取得
- 送信ドメインを設定
- 送信ドメイン ID を取得
- OmniCRM の送信ドメインを設定
- `crm_config.yaml` に送信ドメイン ID を設定

送信内容

送信内容のテンプレート

- `{{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

□□□□

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

□□□

- `failure_list` - Ansible □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□

□□□□

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

□□□

□□□□ PDF □□□□□ Jinja2 HTML □□□

□□□□□ `/OmniCRM-API/invoice_templates/`

□□□□□□□□□□□□□□□□□□ □□ **PDF** □□ □□□

CRM □□ URL

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

□□□

- □ Ansible □□□□□□ API □□
- □□□□□□□□□□□□□□ CRM □□□
- □□□□ API □□□□□□ URL

□□□

- `http://localhost:5000`
- `https://api.yourcompany.com`

□□□

- □□□□□□□□
- □□□□□□□□□□□□□□□□□□□□ URL
- □□□□□□□□

OCS □ CGRates □□

ocs:

```
ocsApi: 'http://10.179.2.142:8080/api'
ocsTenant: 'mnc001.mcc001.3gppnetwork.org'
cgrates: 'localhost:2080'
```

□□□

- `ocsApi` - □□□□□□□ OCS API □□
- `ocsTenant` - □□□ OCS □□□□□□□□
- `cgrates` - CGRates JSON-RPC API □□□□□□□□

□□□ □□□□□□□□□□□□□□□□

SMSC □□□□□□□□

smc:

```
source_msisdn: 'YourCompany'
smc_url: 'http://10.179.2.216/SMSc/'
api_key: 'your_smc_api_key'
```

□□□

- □□□□□ SMS □□□□□□□□□□□□□□□□ 2FA □□□
- `source_msisdn` - □□□□□□□□□□ ID□□□□□□□□□□□□
- `smc_url` - SMSC □□ API □□

- `api_key` - SMSC API キー

環境変数

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

環境変数 CBC API キー

環境変数 `features_cell_broadcast`

JWT

```
jwt_secret: 'CHANGE_ME_ON_DEPLOYMENT' # Ansible 環境
```

環境

- JWT
- -
- /
-

Ansible

Ansible JWT

1. Ansible 64
2. Ansible
3. -

環境

- 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 📜📜

```
stripe:  
  secret_key: 'sk_live_xxxxxxxxxx'  
  publishable_key: 'pk_live_xxxxxxxxxx'  
  currency: 'aud'  
  statement_descriptor_suffix: 'YOURCOMPANY'
```

📜📜

- `secret_key` - Stripe 📜 API 📜📜📜📜📜📜📜
- `publishable_key` - Stripe 📜📜📜📜📜📜📜📜📜
- `currency` - ISO 4217 📜📜📜📜aud📜usd📜gbp📜eur 📜📜
- `statement_descriptor_suffix` - 📜📜📜📜📜📜📜📜📜

📜📜📜📜

- `sk_test_...` と `pk_test_...` の両方
- `sk_live_...` と `pk_live_...` の両方

必要なファイル

- 会社名を "YOURCOMPANY"
- 22 日間
- 必要書類
- PDF ファイル `YOURCOMPANY_12345.pdf`

必要なファイル *（日本語）*

API IP 設定

API IP 設定 API 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 設定 API 設定
ip_whitelist:
  "10.179.2.142":
    roles: ["admin"]
```

設定

- API 設定
- IP 設定
- API 設定
- 必要書類

API 設定

```
openssl rand -base64 48
```

####

- `admin` - #####
- `RBAC` #####

#####

- ##### API ##### 32 #####
- `IP` #####
- #####
- ##### API #####
- ##### API #####

#####

- ##### IP #####
- ##### IP `127.0.0.1:::1`
- ##### API #####
- #####

`administration_api_keys` `concepts_api`

UI #### `crm_config.yaml` #### `ui:`

`omnicrm-ui` ##### API

- ##### `GET /crm/public/ui-config` ##### `ETag` UI `localStorage` #####
- ##### `logos` `favicons` `Web` ##### `ui.assets_dir` #####
`/crm/public/assets/...` #####
- ##### **branding:** ##### `ui.branding` `ui.theme` `UI` #####

`ui:` ##### `OmniCRM-API/crm_config.yaml`

```

ui:
  version: '1'
  assets_dir: '/etc/omnicrm/deployment_assets'
  branding:
    client_name: 'XXXX'
    portal_name: 'Omnitouch'
    self_care_name: 'XXXX'
    logos:
      sm: logos/logoSm.png
      dark: logos/logoDark.png
      light: logos/logoLight.png
      favicon: logos/favicon.ico
  theme:
    primary: '#0D0629'
    secondary: '#CBB42B'
    font_family: Quicksand
  locale:
    default_language: en
    locale: en-GB
    default_country: Australia
  integrations:
    google_maps_api_key: ''
    google_tag_id: ''
    grafana:
      dashboard_ids: 'id1,id2'
      dashboard_labels: 'Ops,Finance'
  features:
    cbc_mode: no
    simplified_addresses: no
  links:
    faqs_url: 'https://example.com/faqs'
    support_url: 'https://example.com/support'
  web_apps:
    - name: GitHub
      url: https://github.com
      icon_path: webapp_icons/github.png

```

XXXX Omnitouch logo [omnicrm-api] XXXXXXXXXX

```

/usr/share/omnicrm/deployment_assets/logos/XXXXXX deployment_assets [
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX /etc/omnicrm/deployment_assets [Ansible [
group_vars/deployment_assets/ [ ] logo_* [ ] ui.branding.logos [ ]
ui.assets_dir: '/etc/omnicrm/deployment_assets' [ ] [ ] /etc/omnicrm [ ]

```

```

##### Docker ##### API ##### UI_ASSETS_DIR ##### /app/deployment_assets ##### GET
/crm/public/assets/ ##### logos/ ##### login/ ##### webapp_icons/ #####

```

```

❯ VITE_* 000 0000000000000000 ui: 0000 .env.example 000000 npm run dev 0
API 0000000000 runtimeConfig.js 00 VITE_*0

```

□□□□ (.env)□□□□□□□□

```
❏❏❏ /OmniCRM/OmniCRM-UI/.env
```

API □□□□□□□□

```
# Google Maps API
VITE_GOOGLE_API_KEY=your_google_api_key
```

```
# npm install --no-browser
```

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

```
# stripe paypal
VITE_PAYMENT_VENDORS="stripe,paypal"
```

```
# Stripe ██████████ Stripe██████
VITE_STRIPE_PUBLISHABLE_KEY=pk test xxxxx
```

```
# PayPal ID PayPal
VITE_PAYPAL_CLIENT_ID=YOUR_PAYPAL_CLIENT_ID
```

□ □ □ □ □ □ □ □

- VITE_STRIPE_PUBLISHABLE_KEY [][] crm_config.yaml [][] stripe.publishable_key [][]
- VITE_PAYPAL_CLIENT_ID [][] crm config.yaml [][] paypal.client id [][]

다음은 VITE_* UI 관련 설정을 정의하는 예시입니다. - [자세히 보기](#)

기본 설정

```
# 기본 설정
VITE_COMPANY_NAME="ShellFone"
VITE_PORTAL_NAME="ShellManager"
VITE_SELF_CARE_NAME="ShellCare"
VITE_COMPANY_TAGLINE="Phones with Shells"
```

UI 관련 설정

- `COMPANY_NAME` - 회사명
- `PORTAL_NAME` - 포털/관리자 페이지명 "ShellManager"
- `SELF_CARE_NAME` - 셀프케어 페이지명 "ShellCare"
- `COMPANY_TAGLINE` - 회사 슬로건

언어 및 지역 설정

```
# 언어 설정
# 기본 언어: en, ch, fr, gr, it, ru, sp
VITE_DEFAULT_LANGUAGE=en
VITE_LOCALE="en-GB"

# 기본 위치 설정
VITE_DEFAULT_LOCATION="Sydney, Australia"
VITE_DEFAULT_COUNTRY="Australia"

# 통화 설정
VITE_CURRENCY_CODE="GBP"
VITE_CURRENCY_SYMBOL="£"
```

언어 코드

- `ar` - 아랍어
- `ch` - 중국어

- `en` - English
- `fr` - French
- `gr` - Greek
- `it` - Italian
- `ru` - Russian
- `sp` - Spanish

Currencies

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

Open `crm_config.yml` and `stripe.currency`

Example

```
# Colors
VITE_PRIMARY_COLOR=#405189

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

Example

- `VITE_PRIMARY_COLOR` - Color
- `VITE_SECONDARY_COLOR` - Color
- `VITE_TERTIARY_COLOR` - Color
- `VITE_SUCCESS_COLOR` - Color #28a745
- `VITE_INFO_COLOR` - Color #17a2b8

- `VITE_WARNING_COLOR` - `ffcc107`
- `VITE_DANGER_COLOR` - `dc3545`
- `VITE_LIGHT_COLOR` - `f8f9fa`
- `VITE_DARK_COLOR` - `343a40`
- `VITE_PRIMARY_DARK_COLOR` - `1a202c`

Hex color code: `#RRGGBB`

Font family

- `Inter`
- `Roboto`
- `Open Sans`
- `Lato`
- `Quicksand`
- `Poppins`
- `Nunito`
- `Montserrat`
- `Work Sans`
- `Source Sans Pro`
- `Raleway`
- `Ubuntu`
- `Josefin Sans`
- `HKGrotesk`
- `Merriweather`
- `Lora`
- `Playfair Display`
- `System`
- `Quicksand`

Font weight

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

Font weight: **OmniCRM-UI**

- `Inter` - `Inter`
- `Roboto` - `Roboto`
- `Open Sans` - `Open Sans`
- `Lato` - `Lato`
- `Quicksand` - `Quicksand`
- `Poppins` - `Poppins`
- `Nunito` - `Nunito`
- `Montserrat` - `Montserrat`
- `Work Sans` - `Work Sans`
- `Source Sans Pro` - `Source Sans Pro`
- `Raleway` - `Raleway`
- `Ubuntu` - `Ubuntu`
- `Josefin Sans` - `Josefin Sans`
- `HKGrotesk` - `HKGrotesk`
- `Merriweather` - `Merriweather`
- `Lora` - `Lora`
- `Playfair Display` - `Playfair Display`
- `System` - `System`
- `Quicksand` - `Quicksand`

Font weight

Font Families

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

Font Families

- Merriweather, Lora, Playfair Display

Font Families

- System - Default font of the operating system

Font Families

Font Families

Font Families

1. Copy the font file from `OmniCRM-UI/src/assets/fonts/your-font-name/`

- Copy WOFF2 font files
- Copy font files 300, 400, 500, 600, 700
- Copy files `your-font-name-300.woff2`, `your-font-name-400.woff2`

2. In `OmniCRM-UI/src/assets/scss/fonts/_google-fonts.scss` add `@font-face`

```
//
// 日本語 - 例
//

@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. `.env` ファイルの作成

```
VITE_FONT_FAMILY=Your Font Name
```

日本語

- 300 - 日本語
- 400 - 日本語
- 500 - 日本語
- 600 - 日本語
- 700 - 日本語

Web 日本語

日本語 6 日本語 Web 日本語

```
# Web 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 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 3-6
# NAME URL ICON_PATH
```

- `VITE_WEB_APP_N_NAME` -
- `VITE_WEB_APP_N_URL` - URL
- `VITE_WEB_APP_N_ICON_PATH` - public/

GitHub Xero Monday.com Gmail MailJet Slack

Grafana

OmniCRM Grafana `grafana_access` Grafana
/grafana OmniCRM

```
# Grafana
# ID - /grafana/d/ID/name URLs
VITE_GRAFANA_DASHBOARD_IDS=abc123,def456,ghi789

# ID
VITE_GRAFANA_DASHBOARD_LABELS=, ,
```

- `VITE_GRAFANA_DASHBOARD_IDS` - URL Grafana ID
- `VITE_GRAFANA_DASHBOARD_LABELS` - ID

ダッシュボード ID

Grafana のダッシュボード URL を取得する。ID は `/grafana/d/abc123/dashboard-name` の ID を `/d/` に置き換えて取得する。 `abc123`

ダッシュボード ID を OmniCRM のダッシュボード ID に置き換えて OmniCRM のダッシュボード ID を取得する。

ダッシュボード URL は `http://your-server/grafana/d/system-overview-123/system-metrics` の ID を取得する。

- ダッシュボード ID `system-overview-123`
- ダッシュボード ID を取得する。

- ID を取得する。
- ダッシュボード ID を取得する。
- ダッシュボード ID `grafana_access` を取得する。
- ダッシュボード ID を UI に取得する。

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 ID を UI に取得する。

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

111

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

[illegible]

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**

111

- 00000000 - 00000000000000000000
- 00000000 - 00000000000000000000
- 00000000 - 0000000000000000 UI

□ □ □ □ □

1. 設定する ID

Google Analytics 4

- Google Analytics
- GA4
- Web
- ID: G-XXXXXXXXXX

Google Tag Manager

- Google Tag Manager
-
- ID: GTM-XXXXXXX

2. .env 設定

```
VITE_GOOGLE_TAG_ID=G-XXXXXXXXXX
```

3. UI 実行

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

4. 実行

-
-
- Google Analytics

実行 -

実行 -

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

// 商品購入
const handleCheckout = () => {
  trackEvent('checkout_initiated', {
    cart_value: 99.99,
    item_count: 3,
    page: '/cart'
  });
  // 商品購入完了
};
```

商品購入 - 商品購入完了

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

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

商品購入 - 商品購入完了

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

// 商品購入完了
trackPageView('/virtual/page-path', '商品購入完了');
```

商品購入完了

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

注意事項

Google Analytics 利用に関する注意事項

- **GDPR**対応 - 欧州連合 (EU) 域内でのデータ処理に適用される
- **CCPA**対応 - 米国カリフォルニア州の消費者プライバシー法
- 同意 - ユーザーが Google Analytics を使用する前に同意を取得する
- **Cookie** 管理 - Cookie を使用してユーザーを識別する
- 匿名化 - Google Analytics の匿名化機能を使用する

インストール

ステップ	作業	コマンド
GA トラッキング コード	GA ID 取得	GA ID は Google Analytics 上で取得する。GTM-XXXXXXX
環境変数 設定	環境変数 ID	<code>.env</code> ファイルに追加
ビルド 実行	UI 環境変数 設定	<code>npm run build</code> コマンドで UI
デバッグ	デバッグ	デバッグ機能を使用
完了	GA トラッキング	完了

注意事項

- 環境変数で指定する
- 環境変数の値 < 50KB を確認
- 環境変数の値を指定
- 例 `VITE_GOOGLE_TAG_ID` を指定

環境変数

- 例 `OmniCRM-UI/src/utils/googleAnalytics.js`
- 例 `OmniCRM-UI/src/utils/usePageTracking.js`
- 例 `App.js` と `index.js` を指定

環境変数の設定

`VITE_TOPUP_PRICE_PER_DAY=10`

例 環境変数の値/環境変数の値

例

- `VITE_TOPUP_PRICE_PER_DAY` - 環境変数の値

例 環境変数の値 `10` を USD 環境変数の値 `$10` を指定

例 環境変数の値 `features_topup_recharge` を指定

環境変数の設定

OmniCRM 環境変数の値/環境変数の値

環境変数の設定

例 `/OmniCRM-UI/src/assets/images/omnitouch/` を指定

- `OmniCRMUI/assets/images/omnitouch/`
- `OmniCRMUI/assets/images/omnitouch/`

2. `OmniCRM`

```
# OmniCRMUI/assets/images/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. `UI`

- `OmniCRMUI/assets/images/omnitouch/logoDark.png`
- `OmniCRMUI/assets/images/omnitouch/logoLight.png`
- `OmniCRMUI/assets/images/omnitouch/logoSm.png`
- `OmniCRMUI/assets/images/omnitouch/logoLight.png`

`OmniCRMUI/assets/images/omnitouch/`

- `OmniCRMUI/assets/images/omnitouch/`
- `OmniCRMUI/assets/images/omnitouch/`
- `OmniCRMUI/assets/images/omnitouch/` PNG
- `OmniCRMUI/assets/images/omnitouch/` DPI `2` `OmniCRMUI/assets/images/omnitouch/`
- `OmniCRMUI/assets/images/omnitouch/`

`OmniCRMUI/assets/images/omnitouch/`

`OmniCRMUI/assets/images/omnitouch/`

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

🔍

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

// 📄
<img src={logoLight} alt="" height="18" />
```

🔍

1. 📄 /OmniCRM-UI/src/assets/images/ 📄 logo-light.png
2. 📄 CSS 📄 .auth-one-bg
3. 📄 authCarousel.js 📄

🔍

```
/* 📄 CSS */
.auth-one-bg {
  background-image: url('/assets/images/custom-auth-bg.jpg');
  background-size: cover;
  background-position: center;
}
```

📄 PDF ❓❓❓

OmniCRM 📄 Jinja2 HTML 📄 WeasyPrint PDF 📄 PDF 📄

PDF 📄

1. 📄
 - 📄 crm_config.yaml 📄 invoice.template_filename 📄
 - 📄 /OmniCRM-API/invoice_templates/ 📄
2. 📄

- 請求IDを返す
- 請求額を返す
- 請求日/月/年を返す

3. 請求書生成

- Jinja2 を使ってHTMLを生成
- Jinja2 のテンプレートで `{{ invoice_number }}` と `{{ total_amount }}` を埋め込む
- 生成したHTMLを `invoice_templates/rendered/` に保存

4. PDF 生成

- WeasyPrint を使ってHTMLをPDFに変換
- PDFのCSSを `invoice_templates/css/` に保存
- PDFのメタデータを埋め込む

5. 保存

- PDFをBase64エンコードして `Invoice_PDF_Cache` に保存
- SHA256ハッシュを生成
- PDFのメタデータを埋め込む

6. 送信

- 請求書のURLを生成
- 請求書のURLをメールで送信

実装

請求書の生成は Jinja2 を使って HTML を生成

テンプレートは `/OmniCRM-`

`API/invoice_templates/yourcompany_invoice_template.html`

生成

```

{
  'invoice_number': 12345,
  'date': '2025-01-04',
  'client': {
    'name': 'John Smith',
    'address': {
      'address_line_1': '123 Main St',
      'city': 'Sydney',
      'state': 'NSW',
      'zip_code': '2000',
      'country': 'Australia'
    }
  },
  'transaction_list': [
    [
      {
        'transaction_id': 1,
        'title': 'XXXX - XX',
        'retail_cost': 30.00,
        'wholesale_cost': 10.00,
        'created': '2025-01-01'
      },
      {
        'transaction_id': 2,
        'title': 'XXXXXXXX - 5GB',
        'retail_cost': 15.00,
        'wholesale_cost': 5.00,
        'created': '2025-01-15'
      }
    ]
  ],
  'total_amount': 45.00,
  'due_date': '2025-01-31',
  'start_date': '2025-01-01',
  'end_date': '2025-01-31',
  'paid': False,
  'void': False
}

```

XXXXXXXX


```

<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>Start Date</strong> {{ start_date }} - {{ end_date
}}</p>
</div>

<div class="customer-details">
  <h3>Customer Information</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>Item</th>
      <th>Description</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: ${{ "%.2f"|format(total_amount) }}</p>
</div>

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

```

```

        {}
    </div>
    {% endif %}

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


```

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

```


- {}
- {}
- {}

###

crm_config.yaml

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

4.

- CRM
- PDF
- invoice_templates/rendered/{invoice_id}.html

5.

```
-- PDF
DELETE FROM Invoice_PDF_Cache;
```

PDF

OmniCRM PDF

- PDF
- PDF
-
- 30

- 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 []

[]

- []
- []
- []/[]

[]

[] (crm_config.yaml)

1. [] OmniCRM-API/crm_config.yaml
2. []
3. [] API []

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

[illegible]

□□ (.env)

1. `OmniCRM-UI/.env`
2. `npm run dev`
3. `npm run build` UI

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

- #### 4. UI 開發 Web 開發

--	--	--	--	--

```
npm start
```

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

- [illegible]

11

- 0000 - 000000000000
- 0000 - 000000000000
- 00000000 - 000000000000
- 0000 - git 000000000000

11

- **Ansible** - 配置管理、应用部署、任务自动化
- **Ansible** - 配置管理、应用部署、任务自动化
- **Grafana** - 监控系统、可视化

11

- 2000 - 2001 UI 2000000000
- 2000 - 2001 2000000000000000
- 2000 - 2001 2000000000
- 2000 - 2001 2000000000000000000000

□□□□□

- 簡單 簡單簡單 UI 簡單簡單
- 簡單 簡單簡單簡單 API/UI 簡單

YAML

- 检查 YAML 格式
- 检查 YAML 文件 `yamllint crm_config.yaml`

Stripe

- Stripe 是什麼
- Stripe 的 API 是什麼

Stripe 的 API

- Stripe 的 API 是什麼
- Stripe 的 API 是什麼 `publishable_key` 是什麼

Mailjet

- Mailjet 是什麼
- Mailjet API 是什麼

PDF

- WeasyPrint 是什麼 `pip install weasyprint`
- 什麼是 `crm_config.yaml`
- `invoice_templates/rendered/` 是什麼 HTML 文件
- API 是什麼 Jinja2 是什麼

PDF 的 API

- 什麼是 `file:///full/path/to/image.png`
- 什麼是 `file:///full/path/to/image.png`
- 什麼是 PNG 和 JPEG 文件
- 什麼是 `file:///full/path/to/image.png`

HTML

- CSS 是什麼
- CSS 的 `flexbox` 和 `grid` 是什麼
- 什麼是 `div`
- 什麼是 `div`

API

- 什麼是 `invalidate_invoice_cache()`

- 環境変数ファイルを作成
 - 環境変数ファイル `Invoice_PDF_Cache` を作成
-

環境変数

OmniCRM の環境変数

環境変数

- ☐ `.env.example` を `.env` にリネーム
- ☐ 環境変数ファイルを作成
- ☐ `CGRates` の環境変数
- ☐ `crm_config.yaml` の環境変数
 - ☐ 環境変数
 - ☐ 環境変数
 - ☐ Mailjet API の環境変数 ID
 - ☐ 環境変数
 - ☐ 環境変数
 - ☐ CRM の URL 環境変数
 - ☐ OCS/CGRates の環境変数
 - ☐ SMSC の環境変数
 - ☐ JWT の環境変数
 - ☐ Stripe の環境変数
 - ☐ API の IP 環境変数

環境変数

- ☐ `OmniCRM-UI/.env.example` を `OmniCRM-UI/.env` にリネーム
- ☐ Google Maps API の環境変数
- ☐ Stripe の環境変数
- ☐ 環境変数
 - ☐ 環境変数
 - ☐ 環境変数

- ☐ 環境構築
- ☐ 認証
- ☐ 顧客管理
 - ☐ 顧客検索
 - ☐ 顧客登録
 - ☐ 顧客更新
 - ☐ 顧客削除
- ☐ 注文管理
 - ☐ 注文検索
 - ☐ 注文登録
 - ☐ 注文更新
 - ☐ 注文削除
- ☐ 売上管理
 - ☐ 売上検索
 - ☐ 売上登録
 - ☐ 売上更新
 - ☐ 売上削除
- ☐ 報告管理
 - ☐ 報告検索
 - ☐ 報告登録
 - ☐ 報告更新
 - ☐ 報告削除
- ☐ 設定管理
 - ☐ Web 設定
 - ☐ URL 設定
 - ☐ Google Analytics/Tag Manager ID 設定

開発

- ☐ 環境構築 `logoSm.png` `logoDark.png` `logoLight.png`
- ☐ 環境構築 `OmniCRM-UI/src/assets/images/omnitouch/`
- ☐ 環境構築 `HTML`
- ☐ 環境構築 `OmniCRM-API/invoice_templates/`
- ☐ 環境構築 `crm_config.yaml` 環境構築
- ☐ 環境構築 PDF 環境構築
- ☐ 環境構築 UI `npm run build`

デプロイ

- ☐ 環境構築
- ☐ 環境構築 JWT 環境構築
- ☐ 環境構築 Stripe 環境構築
- ☐ 環境構築 Mailjet API 環境構築
- ☐ 環境構築
- ☐ 環境構築 API 環境構築 IP 環境構築
- ☐ 環境構築 SSL/TLS 環境構築
- ☐ 環境構築 HTTPS 環境構築
- ☐ 環境構築 CORS 環境構築
- ☐ 環境構築

☐ 数据库数据库

项目

- ☐ 数据库数据库
- ☐ 数据库数据库
- ☐ 数据库数据库数据库数据库
- ☐ 数据库数据库 PDF 文档
- ☐ 数据库数据库Stripe
- ☐ 数据库数据库 2FA
- ☐ 数据库 impersonation 数据库
- ☐ 数据库 OCS 数据库
- ☐ 数据库 ActionPlan 数据库
- ☐ 数据库数据库

项目

- ☐ 数据库MySQL
- ☐ 数据库 CGRates
- ☐ 数据库 OmniCRM API
- ☐ 数据库 OmniCRM UI
- ☐ 数据库nginxtraefik
- ☐ 数据库GrafanaPrometheus
- ☐ 数据库
- ☐ 数据库
- ☐ 数据库

项目

- RBAC 数据库
- 数据库
- Ansible 数据库

- `impersonation`
- `impersonation`
- `impersonation`
- `impersonation impersonation`
- `impersonation`
- `administration_api_keys` - API `impersonation`
- `integrations_mailjet` - `impersonation`
- `concepts_api` - API `impersonation`

OmniCRM

OmniCRM

IP GPON ONT CPE

CPE



inventory_items_list UI Ansible -

OmniCRM

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

SIM

Omnitouch 1000 SIM SIM SIM

#####SIM#####SIM##### - #####
#####

#####SIM#####SIM#####

#####SIM#####

GPON#####

#####GPON#####

#####

#####

#####OmniCRM#####

#####

InventoryTemplate#####

- #####MAC##### itemtext1 ##### itemtext2 #####
- #####
- ##### itemtext1 ##### itemtext2 #####
- #####IP#####

#####20##### itemtext1 itemtext20 #####

itemtext1_label itemtext20_label #####

itemtext1_label #####“MAC”##### itemtext2_label #####“”

#####**InventoryTemplates**#####
#####

#####

inventory_items_list #####
#####

```
inventory_items_list: ["SIM Card", "Mobile Number"]
"SIM Card"
"Mobile Number"
```

-

UI

□ □ □ □ □ □ □ □ □ □

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

□ □ □ □ □

- o `fa-solid fa-sim-card`
- o `inventory_items_list`

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

- ☐ 
- ☐ 

 [illegible]

項目

- 項目**1**の項目名を「項目」
 - 項目名「ICCID」のSIMの「MAC」の項目名を「項目」
- 項目**2**の項目名を「項目」
 - 項目名「IMSI」のSIMの「項目」の項目名を「項目」
- 項目**3-20**の項目名を「項目」
 - 項目名を「項目」
 - 項目名を「項目」

項目

- 項目名を「項目」
- 項目名を「項目」/項目名

4. 項目

項目

項目

1. 項目 → 項目
2. 項目
3. 項目
4. 項目
5. 項目



項目

項目 `itemtext1_label` 項目



項目

項目 `inventory_items_list` 項目

項目

SIM項目

- 名称“SIM卡”
- 单价2.50
- 数量10.00
- 属性1名称“ICCID”
- 属性2名称“IMSI”
- 属性3名称“SIM卡”类型/eSIM卡

物联网卡

- 名称“物联网卡”
- 单价1.00
- 数量0.00
- 属性1名称“卡号”
- 属性2名称“卡密”
- 属性3名称“卡类型”类型/物联网卡

物联网CPE设备

- 名称“物联网CPE”
- 单价250.00
- 数量450.00
- 属性1名称“MAC地址”
- 属性2名称“序列号”
- 属性3名称“设备类型”
- 属性4名称“品牌”
- 属性5名称“型号”

GPON ONT设备

- 名称“GPON ONT”
- 单价45.00
- 数量0.00数量/数量
- 属性1名称“序列号”
- 属性2名称“MAC地址”
- 属性3名称“PON口”

OmniCRM

Ansible

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

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

□□□□□□

- **inventory_id** - 00000000000000000000
- **item** - 00000000000000000000“SIM”“000000”“0000”
- **inventory_template_id** - 00000000000000000000
- **customer_id** - 00000000000000ID000000
- **service_id** - 00000000000000ID000000
- **item_location** - 00000000📍📍
 - 00000000000000000000000000000000
 - 00000000000000IP0000000000000000
- **item_state** - 0000000000000000
 - 0 - 000000000000
 - 000 - 000000000000

- 品牌 - 品牌名称
- 型号 - 设备型号
- 颜色 - 设备颜色
- 尺寸 - 设备尺寸
- 重量 - 设备重量
- 容量 - 设备容量

设备信息

设备信息20

- **itemtext1** - 设备信息1 `itemtext1_label`
 - 设备信息1 "MAC"
 - 设备信息2 "SIM" "ICCID"
 - 设备信息3 "IMEI"
- **itemtext2** **itemtext20** - 设备信息20
 - 设备信息20 "IMEI"
 - 设备信息21 "SIM" "IMSI"
 - 设备信息22 "IMEI"

设备信息1  设备信息2 `itemtext1_label` `itemtext2_label`

设备信息

- **wholesale_cost** - 批发成本
- **retail_cost** - 零售成本
- **sold_date** - 销售日期

设备信息

设备信息CPE/ONT

- **address_line_1** - 地址行1
- **address_line_2** - 地址行2
- **city** - 城市
- **state** - 州

- **zip_code** - ZIP
- **country** -

Web UI

Web UI

- **google_maps_place_id** - Google Maps
- **plus_code** - Google Maps Plus
- **latitude** -
- **longitude** -

-
-
-
-

URL

management_url URL

- **management_url** - URL
 - Web URL `https://192.168.1.1` ONT CPE
 - **eSIM** eSIM LPA
 - LPA:1\$mdp.example.com\$ACTIVATION-CODE-HERE
 - URL

management_url

- 128x128 URL
- 64x64
-

- 管理IP
- **eSIM**管理CRM管理eSIM
- 管理

• **management_username** - 管理用户名

• **management_password** - 管理密码

管理

管理

• **config_content** - 配置内容

- 配置内容
- 配置内容CPE

• **config_file_path** - 配置文件路径

- 配置文件路径
- 配置文件路径

管理

• **inventory_notes** - 库存备注

- 库存
- 库存
- 库存
- 库存
- 库存

• **created** - 创建时间

• **last_modified** - 最后修改时间

管理

1SIM

```
{
  "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"
}
```

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$mdp.example.com$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


```

{
  "inventory_id": 3001,
  "item": "GPON ONT",
  "inventory_template_id": 15,
  "itemtext1": "ALCL12345678",
  "itemtext2": "AA:BB:CC:DD:EE:FF",
  "itemtext3": "OLT-1, PON 3, ONT 42",
  "itemtext4": "Nokia G-010G-A",
  "item_location": "Customer Premises",
  "item_state": "Assigned",
  "customer_id": 321,
  "service_id": 654,
  "address_line_1": "456 Fiber Lane",
  "city": "Melbourne",
  "state": "VIC",
  "zip_code": "3000",
  "country": "Australia",
  "google_maps_place_id": "ChIJ1234567890",
  "plus_code": "4RRH+2C Melbourne VIC",
  "latitude": "-37.8136",
  "longitude": "144.9631",
  "management_url": "https://192.168.1.1",
  "management_username": "admin",
  "config_content": "# ONT Configuration\nwlan-ssid: HomeNetwork\nwlan-password: encrypted...",
  "wholesale_cost": 45.00,
  "retail_cost": 0.00,
  "inventory_notes": "Provisioned 2025-01-20. Optical power: -22dBm"
}

```



management_url 234 UI

- URL LPA
- Web URL

- eSIM 2 eSIM

- 00000000003400000000000000000000000000IP00

00500000000000

```
{
  "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"
}
```

00000000

item_state00000000000000

- 0 → 000 - 000000000000
- 000 → 000 - 00000000000000
- 0 → 0000 - 00000000000000
- 000 → 00 - 00000000RMA000
- 0000 → 00 - 00000000000000
- 0000 → 00 - 00000000000000
- 00/000 → 0 - 0000000000

0000000000000000

- 00000000000000
- 0000000000000000
- 0000000000000000
- 00000000000000

- □□□□□□□□□□

API

API 是应用程序编程接口 (Application Programming Interface) 的缩写，它定义了不同软件系统之间进行通信的规范和标准。

API 通常由以下部分组成：

1. 接口定义 (Interface Definition)

2. 数据格式 (Data Format)

3. 通信协议 (Communication Protocol)

API 的优势

1. 提高开发效率

API 提供了标准化的接口，使得开发人员可以快速集成不同的系统，从而提高了开发效率。

1. 降低开发成本

2. 提高系统可扩展性

3. 促进系统互操作性

◦ 降低开发成本

◦ 提高系统可扩展性

- 國際標準化組織
- 國際標準化組織技術委員會
- 國際標準化組織

國際標準

國際標準化組織

1. 國際標準化組織
2. 國際標準化組織
3. 國際標準化組織
4. 國際標準化組織
 - 國際標準化組織技術委員會
 - 國際標準化組織技術委員會
 - 國際標準化組織技術委員會 URL
 - 國際標準化組織技術委員會
 - 國際標準化組織技術委員會 2099-01-01
5. 國際標準化組織
6. 國際標準化組織

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}
```

□□□□□

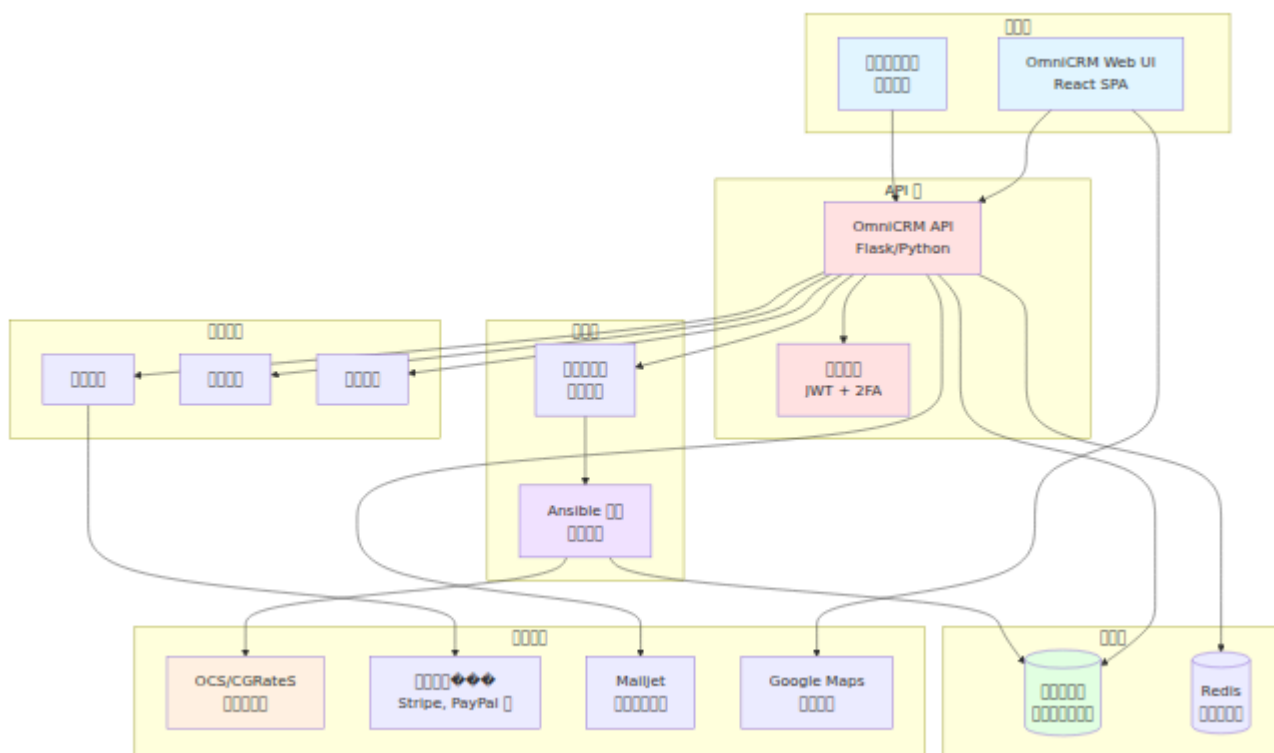
```
DELETE /crm/tag/tag_id/{tag_id}
```


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 フレームワーク
- 静的コンテンツホスティング
- 静的ファイル
- 静的ファイル

API 機能

- 顧客管理
- 商品管理
- 注文管理
- 各種 UI 機能の実装

2. API 実装

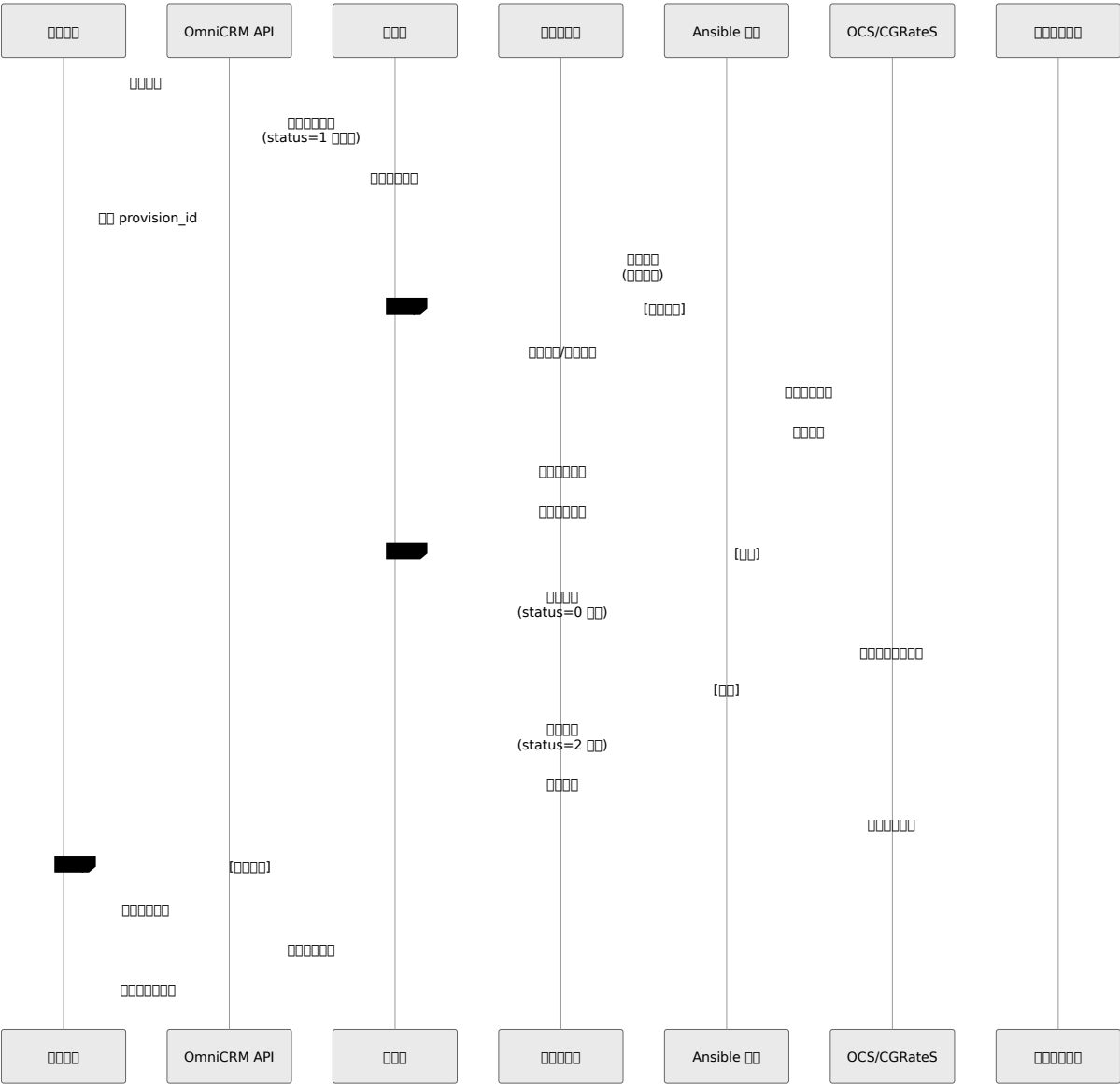
OmniCRM API (Flask/Python)

- 標準的な RESTful API
- OpenAPI/Swagger 仕様
- JWT 認証
- 認証トークン
- WebSocket 通信

API 仕様:

- `/crm/customer/*` - 顧客管理
- `/crm/service/*` - サービス管理
- `/crm/product/*` - 商品管理
- `/crm/provision/*` - 提供管理
- `/crm/transaction/*` - 取引管理
- `/crm/invoice/*` - 請求管理

3. 流程图

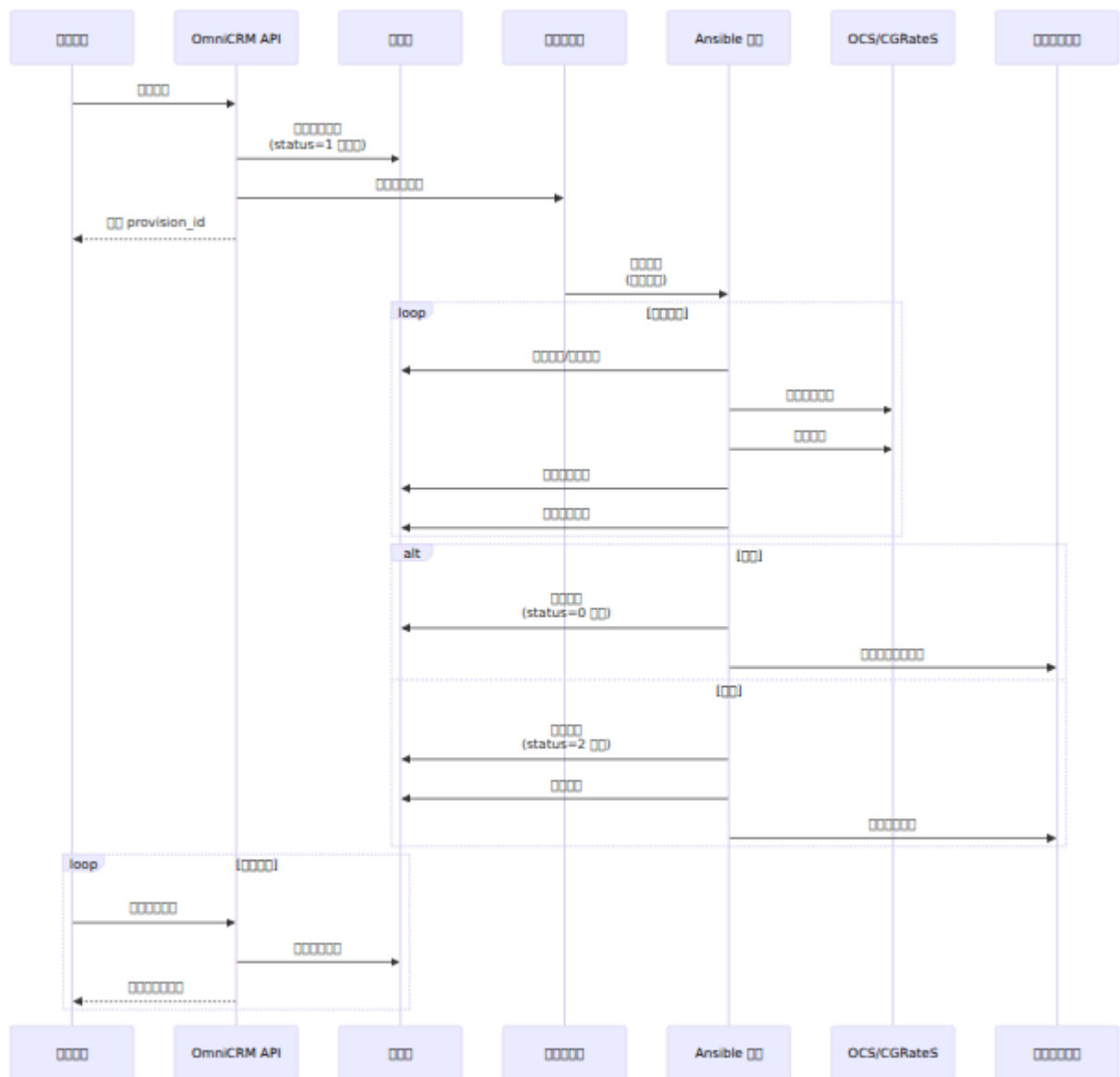


4. 流程图

OCS/CGRateS

-
-
-
-
-

流程图:



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	データ型	制約
customer_id	整数 (PK)	一意
customer_name	文字列	最大 100 文字
customer_account_type	文字列	'個人' または '法人'
customer_status	文字列	'有効', '停止', '凍結', '削除'
customer_payment_type	文字列	'現金' または 'クレジット'
customer_enabled	ブール値	有効/無効
tax_identifier	文字列	法人番号/税務番号
contract_start_date	日付	必須
contract_end_date	日付	必須

注記 - 詳細仕様

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	浮動小数点	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

Table - Table

Field	Type	Comment
service_id	INT (PK)	Service ID
customer_id	INT (FK)	Customer ID
product_id	INT (FK)	Product ID
site_id	INT (FK)	Site ID
service_name	VARCHAR	Service Name
service_uuid	UUID	Service UUID
service_status	INT	Service Status
service_billed	INT	Service Billed
wholesale_cost	DECIMAL	Wholesale Cost
retail_cost	DECIMAL	Retail Cost
bundled_parent	INT (FK)	Bundled Parent

Table - Table

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	'食品', '飲料', '雑貨', 'その他'
provisioning_play	文字列	Ansible プレイブック名
provisioning_json_vars	JSON	Ansible 変数
inventory_items_list	文字列	在庫品リスト
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 - 2024/01/15

欄名	属性	説明
transaction_id	PK (PK)	取引ID
customer_id	FK (FK)	顧客ID
invoice_id	FK (FK)	領収書ID
service_id	FK (FK)	サービスID
title	文字	タイトル
retail_cost	数値	小売価格 = 元/円
payment_type	文字	支払方法 wallet_topup_card, wallet_invoice, postpaid_deferred_charge
wallet_ledger_id	FK (FK)	ウォレット明細ID wallet_ledger
void	文字	キャンセル

備考 - 参照関係

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	データ型	コメント
inventory_id	整数 (PK)	在庫ID
customer_id	整数 (FK)	顧客ID
service_id	整数 (FK)	サービスID
inventory_template_id	整数 (FK)	在庫テンプレートID
item	文字列	在庫品名SIM 在庫品名
item_state	整数	'0', '1000', '1000' 等
itemtext1-20	文字列	在庫品名
management_url	文字列	在庫品 URL
config_content	文字列	在庫品

在庫品 - 在庫品名

欄名	データ型	コメント
inventory_template_id	整数 (PK)	在庫テンプレートID
item	文字列	在庫品名
itemtext1_label	文字列	itemtext1 在庫品名
itemtext2_label	文字列	itemtext2 在庫品名

在庫品

在庫品 - 在庫品名

欄名	型	コメント
provision_id	整数 (PK)	プロビジョニングID
product_id	整数 (FK)	製品ID
customer_id	整数 (FK)	顧客ID
service_id	整数 (FK)	サービスID/サービス名
provisioning_play	文字列	Ansible プレイブック
provisioning_status	整数	0=成功, 1=失敗, 2=中止

Provision_Event - イベント

欄名	型	コメント
provision_event_id	整数 (PK)	イベントID
provision_id	整数 (FK)	プロビジョニングID
event_name	文字列	イベント名
event_number	整数	イベント番号
provisioning_status	整数	0=成功, 1=失敗, 2=中止
provisioning_result_json	JSON	イベント結果

イベント

イベント - イベント

欄名	型	説明
id	整数 (PK)	一意識別子
username	文字列	ユーザー名
email	文字列	Eメールアドレス
email_verified	ブール値	Eメールが確認済みかどうか
is_2fa_enabled	ブール値	2段階認証が有効かどうか
totp_secret	文字列	TOTP 秘密鍵

ユーザー - ユーザー情報

欄名	型	説明
id	整数 (PK)	一意識別子
name	文字列	名前
description	文字列	説明

リソース - リソース情報

欄名	型	説明
id	整数 (PK)	一意識別子
name	文字列	リソース名 (resource.action)
description	文字列	説明

リソース:

- 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

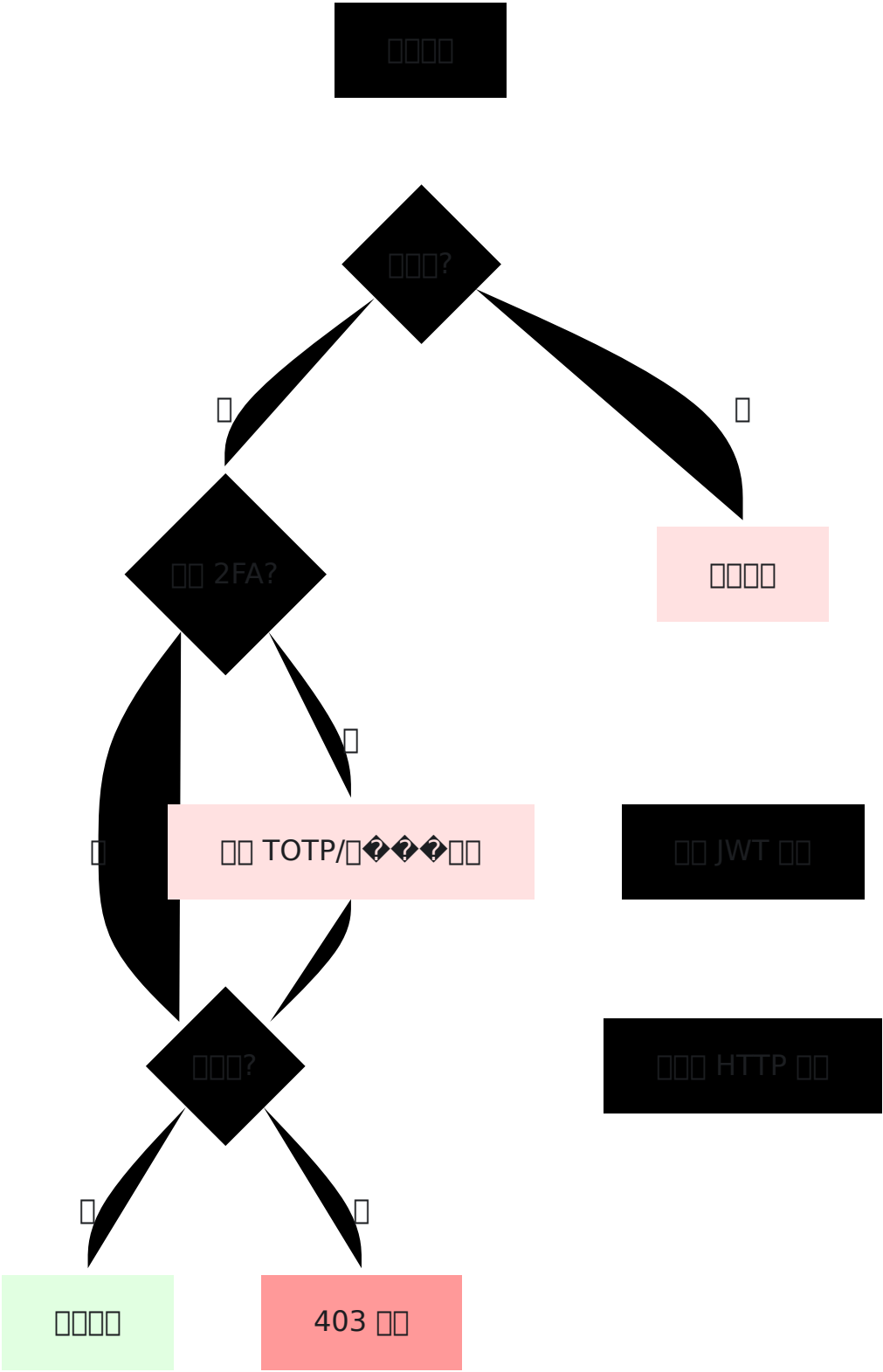
- 國際化國際化
- 國際化國際化國際化國際化
- Plus Code 國際化
- 國際化國際化

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

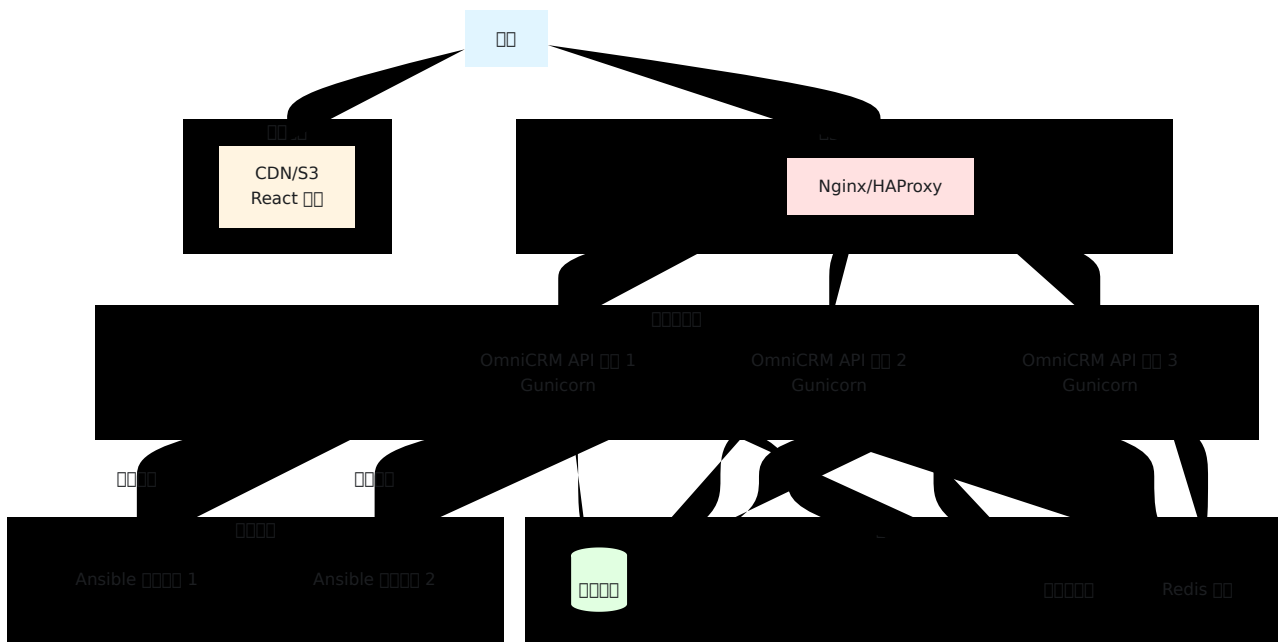


认证成功:

- 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 Web 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
- Google Maps 地图

部署与运维

部署:

- 部署 API 服务
- 部署数据库
- Redis 部署

运维:

- 数据库备份
- 日志
- 性能监控

测试:

- [Docker](#)
- [Ansible](#) [Playbooks](#)
- [Kubernetes](#)
- [Helm](#)

Deployment

OmniCRM [Deployment](#) Prometheus [Monitoring](#) [Alerting](#)

Features:

- API [Endpoints](#)
- [Database](#)/[Cache](#)
- [Authentication](#)
- [Integrations](#) OCS [Stripe](#) [Mailjet](#)
- [Reporting](#)

Stack:

- [Backend](#) [Flask](#)
- [Configuration](#) [Ansible](#) [Playbooks](#)
- [Frontend](#)
- [Database](#)

Endpoints: [/crm/metrics](#) Prometheus [Monitoring](#) [Alerting](#)

Usage

- [Ansible](#) [Playbooks](#) - [Deployment](#)
- [Database](#) - [Setup](#)
- [Frontend](#) - [Setup](#)
- [API](#) [Endpoints](#) - [API](#) [Endpoints](#)
- [RBAC](#) - [Setup](#)

[illegible]

OmniCRM のインストールとアップグレード (2FA)のインストールとアップグレードのインストールとアップグレード
UI のインストール

RBAC

11

OmniCRM

- 密碼/帳號 - 密碼要強
- 密碼 (2FA) - 密碼 TOTP 密碼
- 密碼 - 密碼要強 30 天
- 密碼 - 密碼要強 30 天
- 密碼 - 密碼 2FA 密碼
- 密碼 - 密碼 Google Apple Facebook 密碼
- 密碼 - 密碼要強

--	--	--	--

[illegible]

□ □ □ □

□ □ □ □ ◆ ◆

1. 請將下列各字填入適當的方格中。
2. 請寫出「**三**」的筆順。
3. 請在下列各字後面，用粗體字寫出它的部首。
4. 請寫出「**三**」的意義。

□ □ □ □ □ □ □ □ □

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

□ □ □ □ □ □

30 HTTP-only cookies

00/000000

□□ □□□□ (□) □□□□□□□□

如何 2FA 如何

如何设置 2FA 如何设置/如何 2FA 如何设置

如何设置

1. 如何设置 Google Authenticator 如何 Authy 如何
2. 如何 OmniCRM 如何
3. 如何 6 如何
4. 如何 6 如何
5. 如何设置如何

如何设置

如何设置

1. 如何 “如何” 如何
2. 如何设置如何  如何 3fa5b9c2 如何
3. 如何 “如何”
4. 如何设置如何

如何

如何 “如何” 如何

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

```

VITE_ALLOW_SOCIAL_LOGINS=yes

```

[\[00 Google 00 \]](#) [\[• 00 Apple 00 \]](#) [\[00 00 Facebook 00 \]](#)

□□□□□□□□□□□□□□□□□□□□“□□□□”□□□□□□□□□□□□

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

□□ “□□□□□” □□□□□□□□□□□□□□

□□□□□ (2FA) □□

□□□□ 2FA □□□□□□□□2FA □□□□□□□□□□□□ TOTP□□□□□□□□□□

2FA

□ □ □ □ □ □ □ □ □ □ □ □

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

□□□□□□□□ 2FA □□□2FA □□□□□□□□□□□□□□

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

1111

[] []

□□□□□□□□□□□□□□□□ 2FA □□□□□□□□□□

□ □ □ □ □ □ □ □

[] []

□□□□□

1. 認証アプリのインストール

- iOS → Apple App Store → “Google Authenticator”
- Android → Google Play → “Google Authenticator”
- 代替として Authy → Microsoft Authenticator → 1Password

2. 設定

- 認証アプリを開く
- 右上 “+” をタップ
- “QRコード” を選択
- QRコードを読み取る
- 入力 “OmniCRM” を入力 6 桁のコードを入力

3. 設定

- 設定 → 認証アプリ 8 桁
- 認証アプリのコードを入力
- 認証アプリのコードを入力
- 認証アプリのコードを入力
- 認証 “コード” を入力

4. 設定

- 認証アプリのコードを入力 6 桁
- 認証 “コード”
- 認証 2FA を入力

認証 2FA の設定

認証アプリ

認証 2FA の設定

□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□ “□□□□□”

1. 如何 设置 邮件
2. 如何 “发送邮件”

如何 设置 邮件

- 如何 设置 邮件 的 主题 和 内容
- 如何 设置 邮件 的 发送 时间
- 如何 设置 邮件 的 发送 频率 为 1 次 每天
- 如何 设置 邮件 的 “发送邮件” 按钮

如何 设置 邮件 的 主题 和 内容

如何 设置 邮件 的 发送 时间

如何 设置 邮件 的 发送 频率 为 1 次 每天

如何 设置 邮件 的 “发送邮件” 按钮

OmniCRM

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

1

[]

- 1.
- 2.
3. “ ”

- 8
-

- “ ”
-
-

/

XXXXXXXXXXXXXXXX

[XXXXXXXXXX]

XXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXX

XXXXXXXXXXXXXXXXXX

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 - XXXXXXXXXXXXXXXXX
- XX XX - XXXXXX
- XX XXXX - XXXXXX
- XX XX **2FA** - XXXXX 2FAXXXX 2FA XXXX
- ☒ XXXXXXXX - XXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX

XX **1**XXXXXXXXXXXX (X)

XXXXXXXXXX

XXXXXXXXXXXXXXXXXX

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

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

[] []

2

“ ”

3

John Smith

[]

△ • • •

[]

-
-
-
-

- 1.
2. “ ”
- 3.
- 4.

2FA

1 2FA ()

2FA

00000000

00 200000

00 “00 2FA”

00 3000

00000

John Smith 000000000000000000000000 2FA

00000

- 0000000000000000 2FA 000
- 2FA 0000 (0) 00000000000000
- 0000000000000000 2FA

000000000

000 2FA 00000000

- 0000000000000000
 - 00000000
 - 0000
 - 000000

- 認證碼
2. 認證碼
 3. 認證碼 2FA

認證碼

認證碼

認證碼

認證碼 (✉) 認證碼 認證碼 login_count = 0

認證碼 (✉)

認證碼

認證Bob Wilson (<bob@example.com>)

認證碼 • 認證碼 • 認證碼 • 認證碼 • 認證碼

[認證碼] [認證碼]

認證 “認證碼”

認證碼

認證 Mailjet 認證碼

認證碼 api_crmCommunicationUserWelcome

認證碼

認證碼 (✎) 認證碼

認證碼

認證碼

認證碼

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
- 密码 - 字母/数字/特殊字符
- 密码/手机号

用户名

姓名 手机号 () 手机号

密码

姓名John Smith (<john@example.com>)

⚠ 密码强度

密码强度 • 密码长度 • 2FA 开启 • 密码

密码强度

[姓名] [手机号]

姓名 “姓名” 姓名

姓名

姓名

姓名

姓名

- 密码强度

- 0000000000“0000”
- 000000000000
- 00 2FA 0000000000

2FA 0000

- 00 2FA 0000000000
- 00000000000000000000
- 0000000000000000
- 000000000000000000
- 00000000000000000000000000000000

000000

- 0000000000000000
- 0000000000000000
- 0000000000000000
- 0 OmniCRM 0000000000000000

000000

00000000

- 000 2FA 000000000000
- 0000❖❖0000000000
- 0000000000000000
- 000000 2FA
- 0000000000

000000

- 0000000000000000
- 000000000000000000000000
- 0000000000000000
- 000000000000 - 0000000000000000

2FA 0000

- 2FA 2FA
- 2FA 2FA
- 2FA 2FA
- 2FA 2FA
- 2FA 2FA

2FA

- 2FA 2FA
- 2FA 2FA
- 2FA 2FA
- 2FA 2FA
- 2FA 2FA

2FA

“2FA”

- 2FA 2FA
- 2FA
 - 2FA 2FA
 - 2FA 2FA
 - 2FA 2FA
 - 2FA 2FA

2FA

- 2FA 2FA
- 2FA
 - 2FA 2FA → 2FA → 2FA
 - 2FA 2FA 30 2FA
 - 2FA 2FA
 - 2FA 2FA 2FA 2FA
 - 2FA 2FA 2FA 2FA 2FA

“Cookie”

- Cookie Cookies
- - Cookies
 - Cookies
 - OmniCRM
 -

-
- -
 -
 - 5-10
 - Mailjet
 -

- 1
- -
 -
 -

2FA

-
- -
 -
 -

##

- 密碼 密碼/密碼密碼密碼
- 密碼
 - 密碼密碼
 - 密碼密碼密碼 2FA
 - 密碼密碼密碼
 - 密碼 2FA密碼密碼密碼

密碼“密碼 2FA 密碼”密碼

- 密碼 密碼
- 密碼
 - 密碼密碼密碼
 - 密碼 API 密碼
 - 密碼密碼

密碼密碼密碼

- 密碼 API 密碼密碼
- 密碼
 - 密碼密碼
 - 密碼密碼
 - 密碼 API 密碼密碼
 - 密碼密碼

密碼密碼

密碼密碼

- 密碼密碼密碼
- “密碼”密碼密碼 30 秒
- 密碼密碼 HTTP-only cookiesJavaScript 密碼密碼
- 密碼密碼 cookies 密碼 HTTPS 密碼

密碼密碼

- 密碼密碼密碼密碼密碼

- 2FA API
- RBAC API
- Mailjet API

2FA API

- TOTP API
- 2FA API
- 2FA API
- 2FA API

2FA API

- 2FA API
- 2FA API
- 2FA API
- 2FA API IP

2FA API

- 2fa - 2FA API API
- rbac - RBAC API
- administration_configuration - Mailjet API
- integrations_mailjet - Mailjet API
- customer_care - Mailjet API



0fT52ZvoZBE

OmniCRM

OmniCRM is a cloud-based customer relationship management (CRM) system that helps businesses manage their customer data and interactions across all channels.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

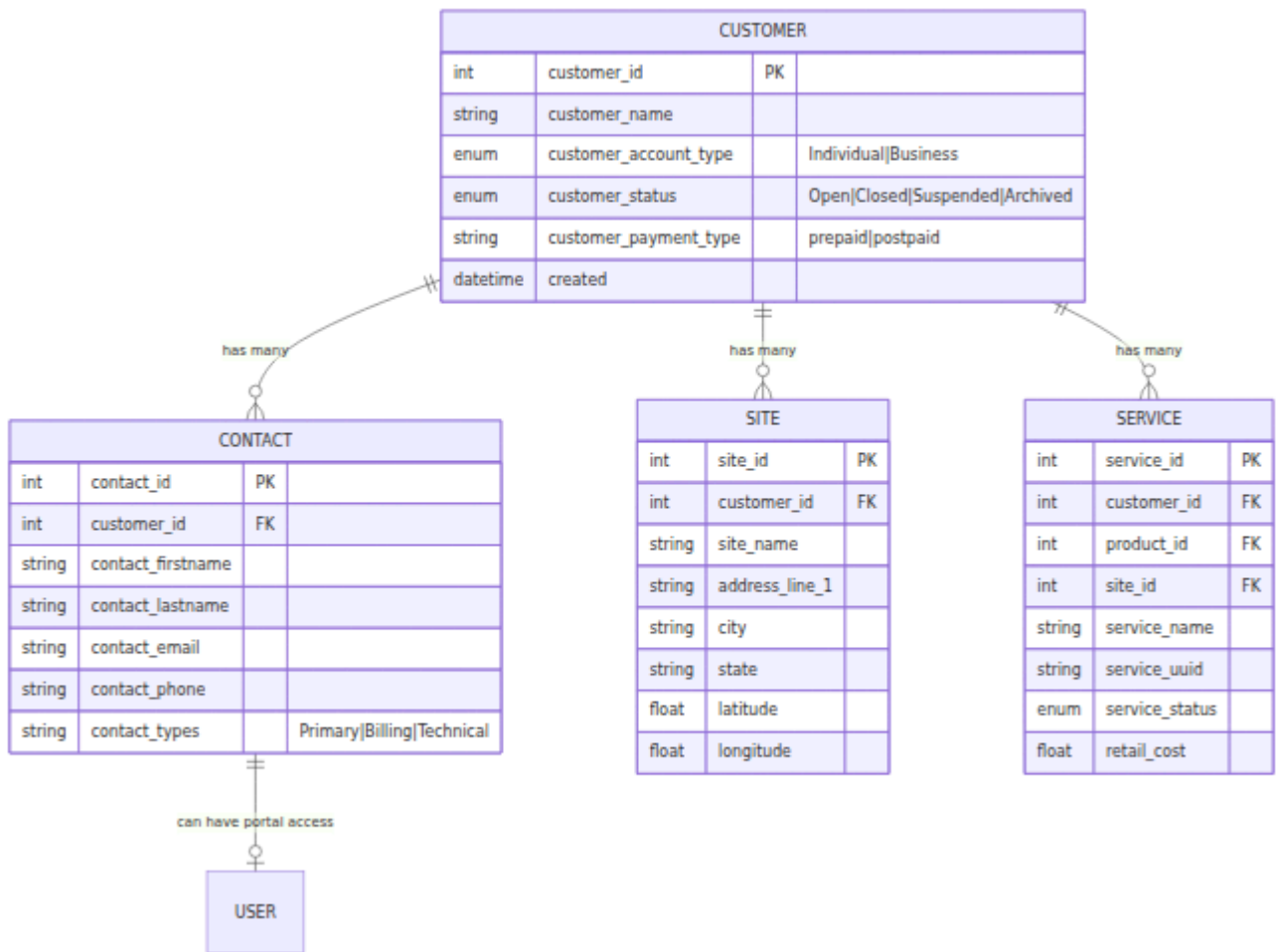
OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

OmniCRM

OmniCRM is a cloud-based CRM system that helps businesses manage their customer data and interactions across all channels. It provides a unified view of the customer journey, from initial contact to post-sale support.

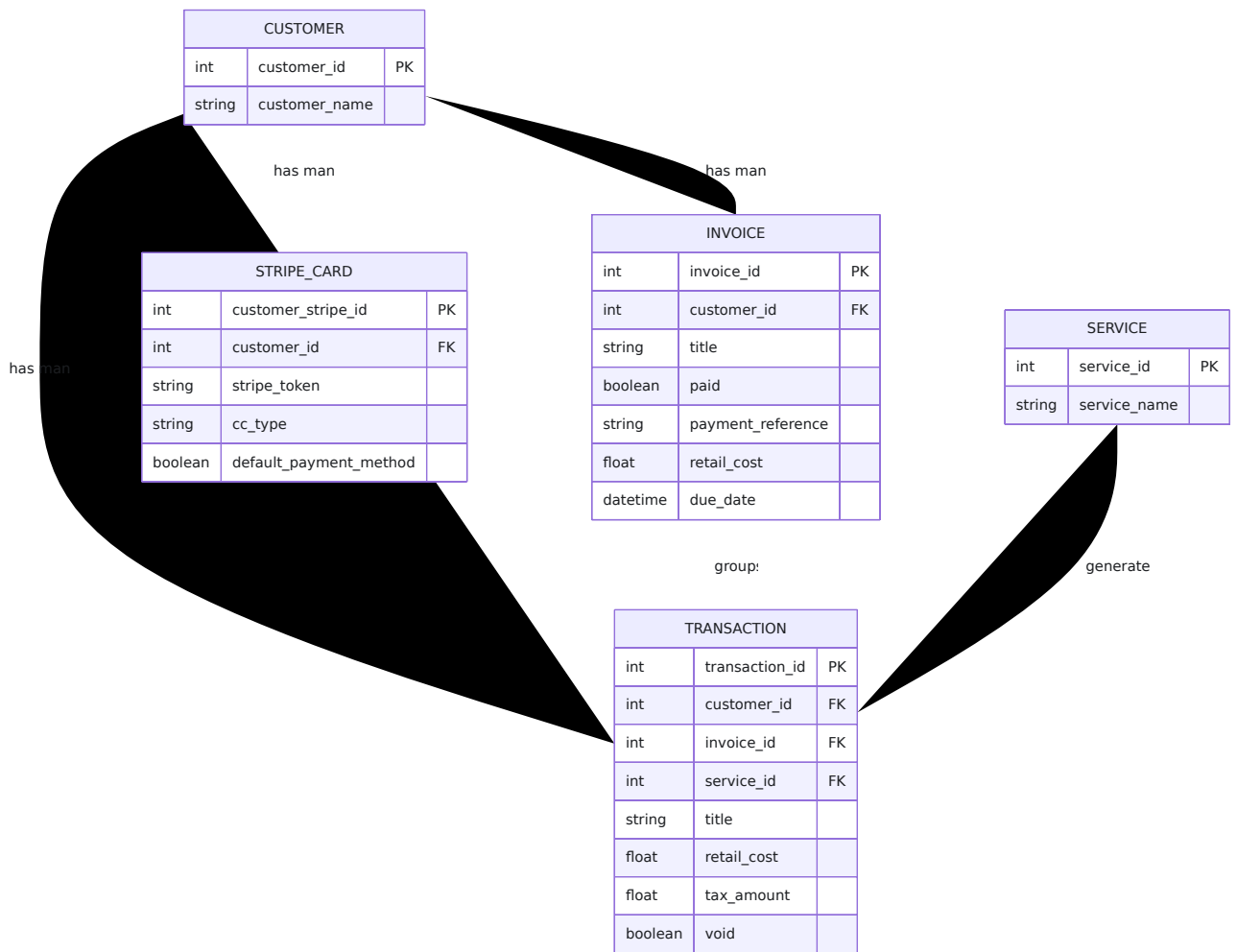


□□□□

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

□□□□□□□□

□□□□□□□□□□□□□□

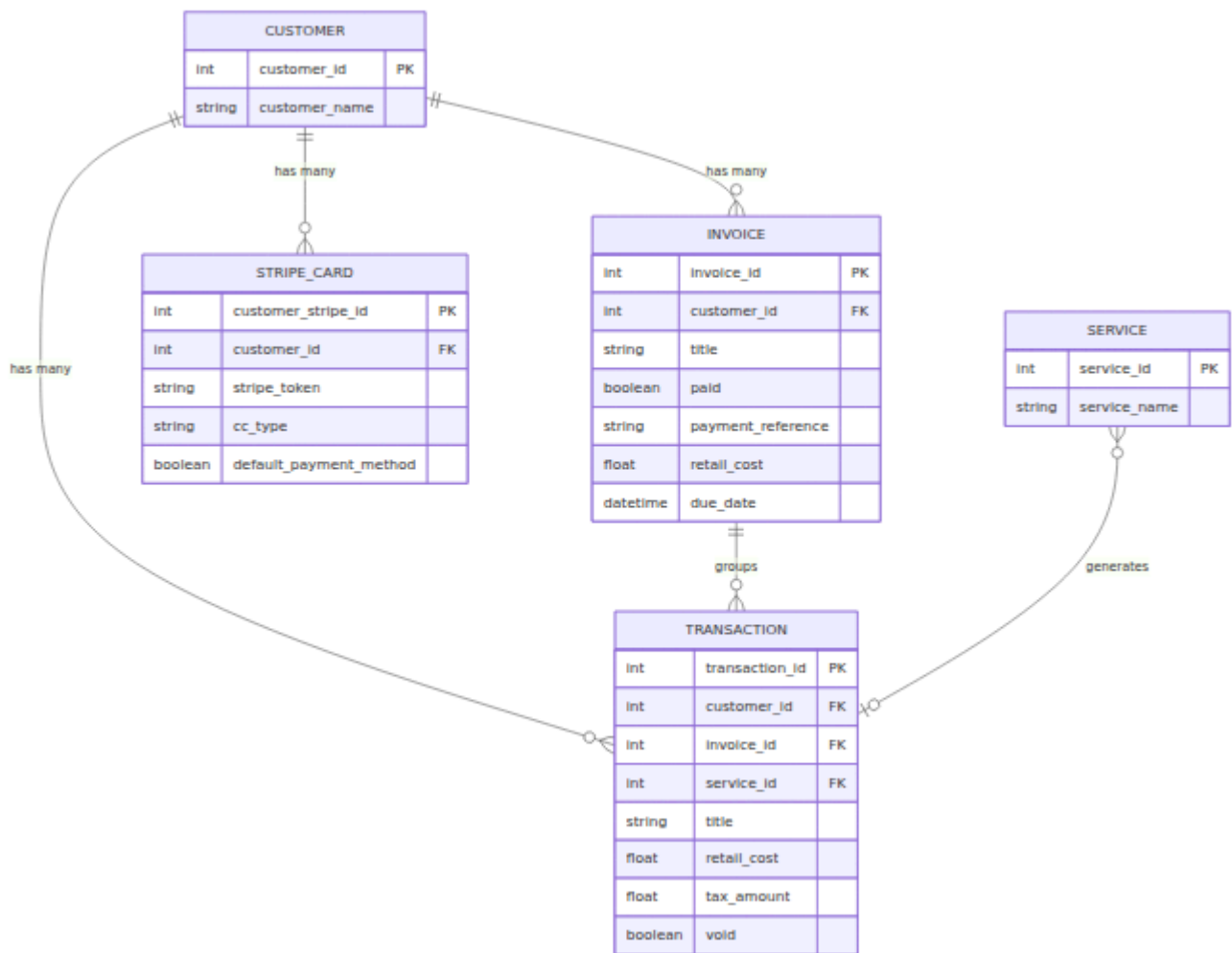


□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- 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

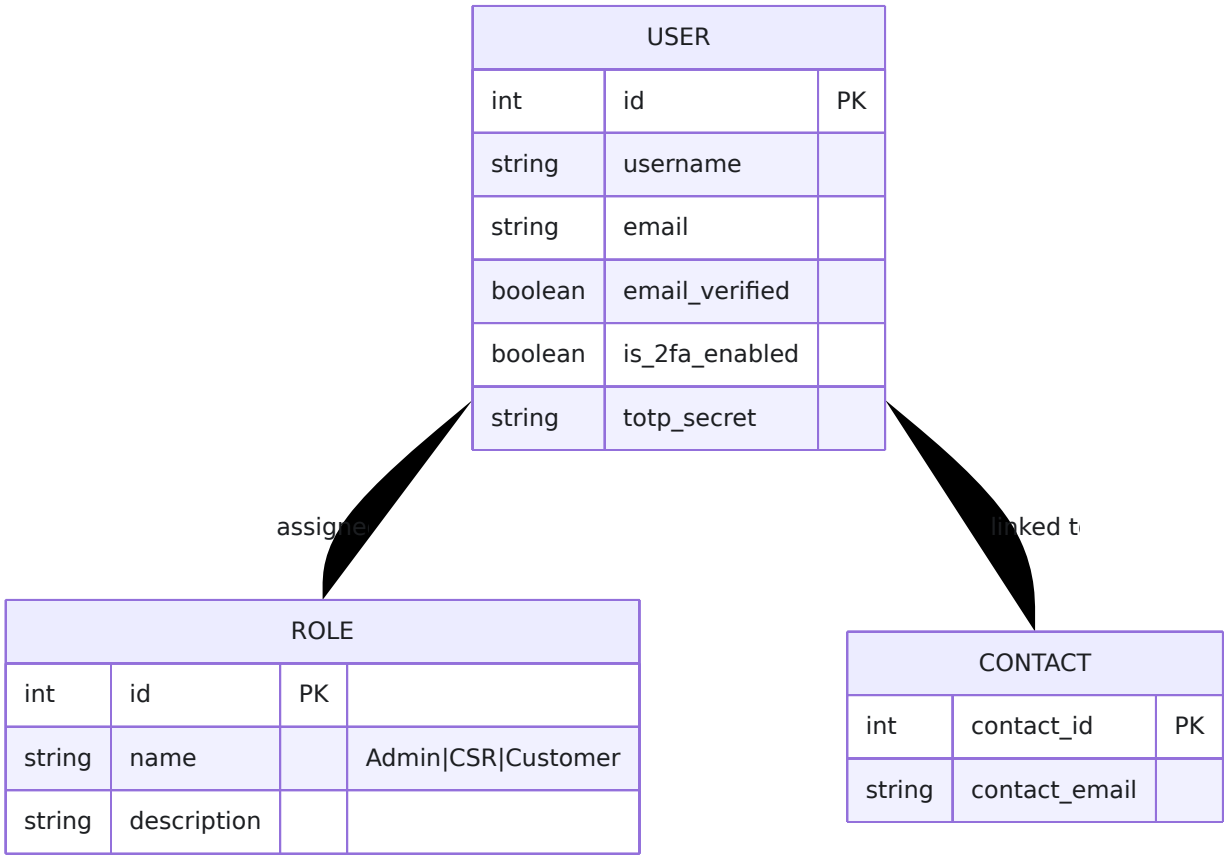
INVENTORY			
int	inventory_id	PK	
int	customer_id	FK	
int	service_id	FK	
int	inventory_template_id	FK	
string	item		
enum	item_state		New Assigned Used Damaged
string	itemtext1		
string	itemtext2		
string	management_url		
text	config_content		

□□□□

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

□□□□□□

□□□□□□□□□□□□□□□□□□



grant:

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

需求

- 系统需要支持用户注册和登录功能
- 系统需要支持用户密码重置功能
- 系统需要支持用户角色管理功能
- 系统需要支持用户权限管理功能

系统

系统需要支持用户注册和登录功能

1

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

OmniCRM

1. □□□□□□

- 000000 - 00000000000000000000
- 0000 - 00000000000000
- 0000 - 0000000000000000

2. □□□□

- 00000000 - 00000000000000000000/0000000000
- 00000 - 00000000000000000000Mailjet
- 00000 - 0000000000000000

000

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

□□□47□□□□□□ □□□□47□□□□□□□□□□

3.

- 0000 - 000000000000
- 000000 - 0000000
-  0000 - 000000000000

4.

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

如何修好你的手机

本文将教你如何修好你的手机，包括更换电池、屏幕、摄像头等。

1. 更换电池

- 准备工具和材料
- 拆卸手机外壳/屏幕
- 取出旧电池并安装新电池

2. 更换屏幕

- 准备工具和材料 `8C3MFJV8+2F`
- 拆卸手机外壳/摄像头
- 拆卸旧屏幕并安装新屏幕

3. 摄像头

- 拆卸“摄像头”螺丝并取出摄像头
- 拆卸“摄像头”螺丝并取出摄像头500
- 拆卸摄像头并安装新摄像头



本文档仅供参考，实际操作时请务必谨慎，并参考相关教程。



OmniCRM ██████████

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □

[illegible]

WebApp Web

□ □

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

--	--	--	--

□ □

□ □

[illegible][illegible]

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □

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

□□□□□□

- 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 - 000000000000

00000000/0000

000000000000

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

00000000

00 "+ 0000" 0000000000

0000000000

00000000/0000

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
- 0000000000000000000000

000000

1. 0000000000
2. 00000000 : 0
3. 00 "00000"
4. 00000000

00000000

- 000000 void = true
- 0000000000000000

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

wallet_ledger_id

- 0000000000000000000000000000
- 00000000 payment_type=wallet_topup_card 0000000000000000
- 0000/000000/000000 0000000000
- 00000000 000000000000000000000000 ? ? ? 000000

钱包ID/钱包ID钱包ID wallet_ledger_id 钱包ID钱包ID void=true 钱包ID
钱包ID钱包ID钱包ID钱包ID钱包ID

API: DELETE /crm/transaction/transaction_id/{transaction_id}    

.md:    

[illegible]

--

[illegible]

4444

- 零售成本 - 零售税率
 - 零售 - 零售税率
 - 零售 - 零售税率
 - 零售 - 零售 $\text{retail_cost} \times (\text{tax_percentage} / 100)$
 - 零售 - 零售 $\$10.00$ (10%)
- 零售/零售 - 零售
 - 零售
 - 零售0%
 - 零售

零售

- 零售10%\$50.00
- 零售 $\$50.00 \times 0.10 = \5.00
- 零售\$5.00 (10%)

零售**NIL**/零售

- 零售0
- 零售0%
- 零售“-”

資料庫設計

資料庫設計

資料庫設計

資料庫設計

- 資料庫設計
- 資料庫設計
- 資料庫
- 資料庫“資料庫”
- 資料庫設計-1

資料庫設計

- 資料庫
- 資料庫設計
- 資料庫ID
- 資料庫✓
- 資料庫

資料庫

資料庫設計

1. 資料庫設計
2. 資料庫設計
3. 資料庫設計
4. 資料庫 `invoice_id` 資料庫
5. 資料庫“資料庫”

資料庫 `payments_invoices` 資料庫設計

14. 問題の答え

問題 3 問題の答え

1. 問題の答え
2. 問題の答え
3. 問題の答え : 問題
4. 問題 "問題"
5. 問題
6. 問題
7. 問題

問題 4 問題の答え

1. 問題
2. 問題 "問題"
3. 問題
4. 問題
5. 問題
6. 問題
7. 問題1-31
8. 問題

問題

問題

- 問題 問題
- 問題 問題問題問題問題問題問題問題問題問題問題
- 問題 問題問題問題問題問題
- 問題 問題問題問題問題問題問題/問題問題問題問題問題問題問題問題問題

- [illegible]

□ □ □ □ □ □

- 
- 

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

- [illegible]

□ □ □ □ □ □ □ □ □ □

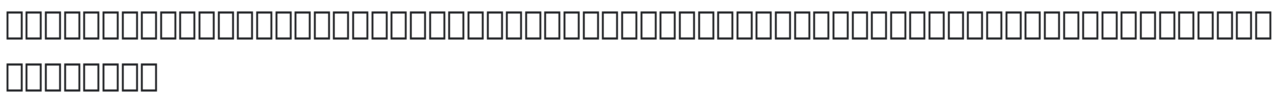
- 1000 0000000000000000
- 1000 000000“000000”0000000000000000

□ □ □ □ □ □ □ □ □ □ □ □ □ □

- [illegible]

□ □ □ □

- `payments_invoices` - 1000000000
- `payments_process` - 1000000000
- `basics_payment` - 1000000000
- `csa_activity_log` - 1000000000000000



OmniCRM API

OmniCRM API HTML templates using Jinja2

Invoice Templates

OmniCRM API invoice templates/

Templates

- sample_invoice_template.html - Sample invoice template
- cifi_invoice_template.html - CIFI invoice template

Example

OmniCRM API crm_config.yaml

```
invoice:
  template_filename: 'sample_invoice_template.html'
```

Invoice Data

Invoice data using Jinja2

Variables

- {{ invoice_number }} - Invoice ID INV-2025-001234
- {{ date }} - Invoice date 2025-01-10T12:00:00
- {{ due_date }} - Due date 2025-02-10
- {{ start_date }} - Start date
- {{ end_date }} - End date
- {{ total_amount }} - Total amount
- {{ total_tax }} - Total tax

Example

- {{ client.name }} - Client name

- `{{ 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

00000

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 %}
```

00000

- `sub_transaction.transaction_id` - 00ID
- `sub_transaction.created` - 0000/00
- `sub_transaction.title` - 0000
- `sub_transaction.description` - 0000
- `sub_transaction.retail_cost` - 00000
- `sub_transaction.tax_percentage` - 0000000000100010%0
- `sub_transaction.tax_amount` - 000000000

000000000

```
<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 sample_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>Your Company</p>
  <p>123 Main Street</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</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. 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><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，并指定了 `tax_percentage` 为 0%。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并指定了 `tax_percentage` 为 0%。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并指定了 `tax_percentage` 为 0%。

```

<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 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. □□□□□□□
 - □□□□□□□□□□
 - □□□□□□□□□□

- 00000000
- 0000000000

3. PDF 0000

- 00000000<5MB
- 0000
- 0000000000
- 0000000000

4. 000000

- 00000000
- 0000
- 0000000000

5. 00000000

- PDF 00000000
- 000000 Mailjet 00015MB
- 0 Gmail0Outlook0Apple Mail 000

000000000000

000000 API 00000000

```
curl -X GET "http://localhost:5000/crm/invoice/{invoice_id}/pdf" \
-H "Authorization: Bearer YOUR_TOKEN" \
--output test_invoice.pdf
```

00000000

00000000

- 000 0000000000000000
- 000 0000000000000000000000000000

PDF 000000

- 000 0000 CSS 00
- 000 00 CSS 2.1 000000 WeasyPrint 000 CSS

- 2. 查詢 - 查詢所有發票PDF
 - 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 - 查詢所有發票查詢 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 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

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

📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

- 📄📄 - 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

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

□□□□□

- 1,000 000000 "k" 000000\$1.5k
- 1,000,000 000000 "M" 000000\$2.3M
- 1,000,000,000 000000 "B" 000000\$1.1B

□□□□□□

- "00" e "01" 000000000000
- 0000000000000000
- 0000000000000000 000000000000
- 000000000000

--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- **ID** - ID
- - /
- - "N/A"
- -
- -
- -
- -
- -

- ↓ () - PDF
- () -
- () -
- ☒ () -
- () - Stripe Stripe

"+"

##

- 00000 - 00000000000000000000000000000000
- 00 - 0000/0000000000 "00 [00]"0
- 0000 - 000000000000 14 000
- 0000 - 0000000000000000
- 0000 - 0000000000000000
- 0000 - 0000000000000000/000000

□ □ □ □ □

- ✓ (0000) - 000000000000
- × (000) - 000000000000
- 00 - 0000000000
- 0000 - 000000
- 00000000000000000000
- 00/0000000000000000

□□□□□□

1. 00000000000000000000
2. 0000000000000000/000000
3. 0000000000000000000000
4. 00000000000000000000
5. 0000 PDF 000000
6. 000000000000 invoice_id 000000
7. 000000000000000000000000
8. 000000000 "000" 00

□□□□□

[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 (↓) 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
 - 결제시스템의 종류
 - 결제시스템의 종류
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. Mailjet の `api_crmCommunicationCustomerInvoice` を呼び出す
4. 請求書データを取得する
 - 顧客の PDF 番号
 - 請求書番号
 - 請求書日付
 - 支払期日
 - 請求書総額
 - 請求書 URL
5. 請求書データを Mailjet に送信する

請求書データを取得する

請求書データを取得する "請求書" を取得する

請求書データを取得する

- `{{ var:customer_name }}` - 顧客名
- `{{ var:invoice_number }}` - 請求書 ID
- `{{ var:invoice_date }}` - 請求書日付
- `{{ var:due_date }}` - 支払期日
- `{{ var:total_amount }}` - 請求書総額
- `{{ var:invoice_url }}` - 請求書 PDF URL
- `{{ var:pay_url }}` - 請求書支払 URL

請求書データを取得する

- 請求書データを取得する - `crm_config.yaml` の Mailjet API キー
- 請求書データを取得する - 請求書データを取得する
- **PDF** 請求書 - 請求書 PDF を生成する

請求書データを取得する

請求書データを取得する () 請求書データを取得する

請求書データを取得する

-  
- 

5/5

1. 〇〇〇〇〇〇〇〇〇〇〇〇
2. 〇〇〇〇〇〇 ()
3. 〇〇〇〇〇〇〇〇

□□□□□□

- `void = true`
- `invoice_id` `null`
- `"000"`
-
- `"000"`
-
- `"00"`

□□□□□

- 00000000
- 00 = "00000000"00000000
- 00 = "0000000000"00000000

退款

如何退款 (一) 退款流程

退款

- 退款流程
- 退款流程 **Stripe** 接口
- 退款流程 `payment_reference` Stripe 接口 ID
- 退款流程

退款

1. 退款流程 Stripe 接口
2. 退款流程 (一)
3. 退款流程

4. 退款 "退款"

5. 退款 Stripe 接口

- 退款 Stripe API 接口
- 退款 Stripe 接口

- 退款 `refund_reference`

6. 退款流程 "退款"

7. 退款

退款

- 0000000000000000
- 000000 "0000"
- 0000000000
- 00000000000000003-7 000000
- Stripe 000000000000

000000

- 0000000000POS 00000000000000000000
- 000000000000000000
- 00000000

0000000000

00

00000000000000000000

- 00 ID
- 00000
- 0000000000000000

00

0000000000000000

000000

- 00000 - 000000000000
- 00000 - 000000000000

000000

- 00000 - 00000000000000
- 00000 - 0000000000000000

- 00/00001010031000000000
 - 00000000
 - 00000000000000
 - 00 **X** 00000000
 - 000000000000
 - 00 "0000"
5. 00000000000000000000
- 0000 "+ 00000000"
 - 0000"0000 - 2025010"
 - 00000000000000000000
 - 00 "00" 000000
 - 00 "0000"
6. 00000000000000000000

0000 3000000000000000

1. 00000000
2. 000000 00 00
3. 00 "+ 00000000"
4. 0000000000
5. 00000000200000
6. 000000000000000000000000
7. 00000000
 - 000000000000"00000000"0
 - 00 **X** 000000
 - 000000000000"000"0
 - 00 **X** 000000
 - 00000000"0000180000"
 - 0000000000
8. 00000000000000000000
9. 00 "000000**18**0"
10. 000000000000000000
11. 00/000000000000000000000000

📌 4. 📌 📌 📌 📌 📌 📌 📌

1. 📌 📌 📌 📌 📌
2. 📌 📌 📌 📌 📌 📌 📌
3. 📌 📌 📌 📌 📌 📌
 - 📌 📌 📌 📌 📌 📌 📌
 - 📌 **X** 📌 📌 📌
 - 📌 📌 📌 📌 📌 📌 📌 📌 📌 📌
 - 📌 **X** 📌 📌 📌 📌
4. 📌 📌 📌 📌
5. 📌 📌 📌 📌 📌 📌 📌
6. 📌 "📌 📌 📌 📌" 📌 📌 📌 📌 📌 📌 📌
7. 📌 📌 📌 📌 📌 📌 / 📌 📌 📌 📌 📌
8. 📌 📌 📌 📌 📌 📌 📌 📌 📌 📌 📌 📌

📌 📌 📌 5. 📌 📌 📌 📌 📌 📌

1. 📌 📌 📌 📌 📌 📌
2. 📌 📌 📌 📌 📌 📌
3. 📌 📌 📌 📌 📌 📌
4. 📌 "+ 📌 📌 📌 📌 📌"
5. 📌 📌 📌
 - 📌 📌 📌 📌 📌
 - 📌 📌 📌 📌 📌
 - 📌 📌 📌 📌 + 7 📌
 - 📌 📌 "📌 📌 📌 📌 📌"
6. 📌 📌 📌 📌 📌 📌 📌 📌
7. 📌 📌 📌 📌 📌 📌 📌
8. 📌 **X** 📌 📌 📌 📌 📌 📌 📌 📌 📌 📌
9. 📌 "📌 📌 📌 📌"
10. 📌 📌 📌 📌 📌 📌 📌 📌 📌
11. 📌 📌 📌 Stripe 📌 📌 📌

第 6 章 数据库系统

1. 数据库系统概述
2. 数据库系统的组成
3. 数据库系统的层次结构
4. 数据库系统的功能
 - 数据库系统 ID 管理
 - 数据库系统用户管理
 - 数据库系统安全管理
 - 数据库系统备份与恢复
5. 数据库系统的性能优化
6. 数据库系统 PDF 文档管理 "PDF" 文档
7. 数据库系统文档管理 "文档" 文档
8. 数据库系统管理

第 7 章 数据库应用

1. 数据库应用概述
2. 数据库应用系统组成
3. 数据库应用系统开发流程
4. 数据库应用系统开发工具
5. 数据库应用系统开发环境
6. 数据库应用系统开发 "应用" 文档
7. 数据库应用系统开发
8. 数据库应用系统开发
9. 数据库应用系统开发
10. 数据库应用系统开发
 - 数据库应用系统开发
 - 数据库应用系统开发
11. 数据库应用系统开发

第 8 章 数据库应用

1. 数据库应用概述

2. 000000 00 00
3. 0000000000
4. 000000000000000000
5. 00000000
6. 0000000000 "0000" 00
7. 00 "00" 0000
8. 000000"00000 2025-01-15"
9. 00 "0000"
10. 00000000000000"000"
11. 000000000000
12. 00000000000"000"

□□□□ **9**□□□□□□□□

1. 创建数据库表
2. 创建 Stripe 账户
3. 创建产品
4. 创建价格
5. 创建客户
6. 创建 "订阅" 类型的 Stripe 订阅
7. 支付
8. 创建 Stripe 支付
9. 创建 "支付" 类型的 Stripe 支付
10. 创建 3-7 天的 Stripe 支付
11. 创建 Stripe 支付

--	--	--	--

□□□□□□ - □□□□□□

- [illegible]

PDF

- 透過 WeasyPrint 將 HTML 轉為 PDF
- 透過 Stripe API 進行付款

Stripe 設定

- 透過 Stripe API 進行付款
- 透過 Stripe 進行付款

Stripe 設定

- 透過 Stripe API 進行付款
- 透過 Stripe 進行付款

Stripe 設定

- 透過 Mailjet API 進行郵件發送
- 透過 Mailjet API 進行郵件發送

Stripe 設定

- 透過 Stripe API 進行付款
- 透過 Stripe 進行付款

PDF 設定

- 透過 PDF 進行付款
- 透過 PDF 進行付款

PDF 設定

- 透過 PDF 進行付款
- 透過 PDF 進行付款

PDF 設定

- 透過 PDF 進行付款
- 透過 PDF 進行付款

PDF 設定

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `getAccountDetails` - 获取账户信息
- `getAccountDetails` - 获取账户信息

账户信息

- `integrations_mailjet` - 获取账户信息
- `administration_configuration` - 获取账户信息
- `payments_transaction` - 获取账户信息

- `payments_process` - 付款处理
- `payment_system_guide` - 支付API使用指南

OmniCRM

OmniCRM 是一個整合了 Stripe 和 PayPal 的 CRM 系統，可以幫助您管理客戶關係，並提供多種支付選項。

OmniCRM 提供多種支付選項，包括：

Stripe

OmniCRM 提供多種支付選項，包括：

- 信用卡 - 支持 Stripe 和 PayPal 的信用卡支付
- 銀行卡 - 支持 Stripe 和 PayPal 的銀行卡支付
- 支票 - 支持 Stripe 和 PayPal 的支票支付
- 現金 - 支持 Stripe 和 PayPal 的現金支付
- 銀行轉帳 - 支持 Stripe 和 PayPal 的銀行轉帳支付
- 銀行卡 - 支持 Stripe 和 PayPal 的銀行卡支付
- 銀行卡 - 支持 Stripe 和 PayPal 的銀行卡支付
- 銀行卡 - 支持 Stripe 和 PayPal 的銀行卡支付

OmniCRM 提供多種支付選項，包括：

- 信用卡 - 支持 Visa、Mastercard、American Express、Discover
- 銀行卡
- 支票

OmniCRM 提供多種支付選項，包括：

OmniCRM 提供多種支付選項，包括：

- stripe、paypal
- Visa、Mastercard
- 支票
- 銀行卡
- 銀行卡
- 銀行卡

အခြေခံအားဖြည့်ဆည်းပေးသည့်စာရွက်

အသုံးပြုသူအား

1. အသုံးပြုသူအား → **အသုံးပြုသူ**
2. အသုံးပြုသူအား
3. အသုံးပြုသူအား

အသုံးပြုသူ

အသုံးပြုသူအား

အသုံးပြုသူအား

အသုံးပြုသူအား 60 အသုံးပြုသူအား

အခြေခံအားဖြည့်ဆည်းပေးသည့်စာရွက်

အသုံးပြုသူအား

အသုံးပြုသူ

- အသုံးပြုသူအား “အသုံးပြုသူ” “အသုံးပြုသူ Visa”
- အသုံးပြုသူအား
- အသုံးပြုသူအား 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

- [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 Stripe Elements PayPal SDK 如何集成支付

如何集成支付

- [API - 如何集成支付](#) CVC Stripe 如何集成支付
- [API - 如何集成支付](#)
- [API - 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iframe項目Stripe Elements / PayPal SDK
- OmniCRM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CVC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

XXXXXXXXXX

XXXX XX → XXXXXXX XXXXXXXXXXXXXXX

XX X XXXX XXXX XX·XXXX Visa ****1234 02/2025 12** XX **Acme Corp MC5678**
03/2025 45 XX XX·X XXXX**9012 01/2025 XXXX XX

XXXX

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

XXX

- XX - XXXXXXXXXXXXXXX
- XX - XXXXXXXXXXXXXXXMailjet

XXXX

XXXXMailjetXXXXXXXXXXXX

- XXX**60** - XXXX

- 30日 - 30日
- 7日 - 7日
- 日 - 日

30日 - 30日

7日 - 7日

Mailjet

- 日
- 日
- 日
- 日

integrations_mailjet

日

日

1. 日
2. 日
3. 日 → 日
4. 日 "日"
5. 日
6. 日
7. 日/日

日

日

1. 日
2. 日 → 日
3. 日
4. 日

5. 資料提供

6. 資料提供



資料提供に関する注意事項

- 資料提供に関する注意事項
- 資料提供に関する注意事項
- 資料提供に関する注意事項

資料提供に関する注意事項

資料提供に関する注意事項

資料提供

1. 資料提供

- 資料提供に関する注意事項
- 資料提供に関する注意事項
- 資料提供
- 資料提供に関する注意事項

2. 資料提供

- 資料提供
- 資料提供に関する注意事項
- 資料“資料提供”に関する注意事項

3. 資料提供

- 資料提供 - 資料提供に関する注意事項
- 資料提供 - 資料提供に関する注意事項
- 資料提供 - 資料提供に関する注意事項

資料提供

1-3 資料提供

- 資料提供
- 資料提供に関する注意事項

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": "Personal Card",
  "is_default": true
}
```

OmniCRM銀行振込

- 銀行振込
- CVV/CVC
- 銀行振込
- PIN

PCI

PCI DSS (Payment Card Industry Data Security Standard)

- PCI DSS - Payment Card Industry Data Security Standard
- PCI DSS - Payment Card Industry Data Security Standard
- PCI DSS - Payment Card Industry Data Security Standard
- PCI DSS - Payment Card Industry Data Security Standard

PCI DSS (Payment Card Industry Data Security Standard)

PCI DSS

PCI DSS 1.2.1

PCI DSS 1.2.1

1. PCI DSS 1.2.1
2. PCI DSS 1.2.1
3. PCI DSS 1.2.1
4. PCI DSS 1.2.1
5. PCI DSS 1.2.1
6. PCI DSS 1.2.1
7. PCI DSS 1.2.1
8. PCI DSS 1.2.1

PCI DSS 2.2.1

PCI DSS 2.2.1

1. PCI DSS 2.2.1
2. PCI DSS 2.2.1
3. PCI DSS 2.2.1 → PCI DSS 2.2.1
4. PCI DSS 2.2.1 2025 12 31
5. PCI DSS 2.2.1

6. 2028년 12월
- 7.
- 8.
- 9.

□□□□**3**□□□□□□□□□□

□□□ □□□□□□□□“□□□□□□□□”

1. □□□□□□□□
2. □□□□□□□□□□
3. □□□□□□□□□□2025□01□□□
4. □□□“□□□□□□□□□□□□”
5. □□□□□□□□□□□□□□
6. □□□□□□□ → 【□□】 → □□
7. □□**“□□□□□□”**
8. □□□□□□□□□□□□□□
9. □□□□□□□□
10. □□□□□□
11. □□□□□□□□
12. □□□□□□“□□□□□□□□□□□□”

□□□□4□□□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

1. 0000000000000000Visa000011110
2. 0000000000000000
3. 00000000Mastercard00002222000000
4. 00000000Amex0000333300000000
5. 000000000000000000000000Mastercard
6. 00Visa000000000000

📄 5 📄 📄 📄 📄 📄 📄 📄 📄 📄

📄 📄 📄

1. 📄 📄 📄 → 📄 📄 📄
2. 📄 “📄 📄 30”
3. 📄 15 📄 📄 📄 📄 📄
4. 📄 → “📄 📄 📄 📄 📄”
5. Mailjet 📄 📄 📄 📄 📄 📄 📄
6. 📄 📄 📄 📄 📄 📄
7. 📄 📄 📄 📄 📄
8. 📄 📄 📄 📄 📄 📄
9. 📄 📄 📄 📄 📄

📄 📄 📄

📄 📄 📄 “📄 📄”

- 📄 Stripe 📄 📄 📄 📄 📄 📄 📄 📄 📄 📄
- 📄 📄
 - 📄 📄 📄
 - 📄 📄 📄 📄 📄
 - 📄 📄 📄 📄 📄
 - 📄 📄 📄 📄 📄 📄 📄 📄

“📄 📄 📄 📄 📄” 📄 📄 📄

- 📄 📄 📄 API 📄 📄 📄 📄
- 📄 📄
 - 📄 📄 📄 📄
 - 📄 📄 📄 📄
 - 📄 📄 📄 📄 📄 📄 📄 📄 📄
 - 📄 📄 📄 📄 📄 📄 📄 📄
 - 📄 📄 📄 📄 📄 📄 📄 📄

プラットフォームの構築

- 言語 選択可能な言語
- フレームワーク
 - 静的型付け言語
 - 動的型付け言語

プラットフォームの“拡張性”評価

- 言語 選択可能な言語
- フレームワーク
 - 静的型付け
 - 動的型付け言語
 - 静的型付け言語

プラットフォーム

- 言語 選択可能な言語
- フレームワーク
 - 静的型付け言語
 - 動的型付け言語
 - 静的型付け言語

プラットフォーム

- 言語 選択可能なSDK/API
- フレームワーク
 - 静的型付け言語
 - 動的型付け言語
 - 静的型付け言語
 - 静的型付けSDK/API
 - 静的型付け/動的型付け

プラットフォーム

- 言語 Mailjet
- フレームワーク

- 編輯 `crm_config.yaml` 增加 Mailjet 設定
- 測試發送郵件
- 測試發送簡訊
- 增加 Mailjet 測試數據

測試

測試

- 測試發送郵件
- 測試 2 個測試數據
- 測試 30 個測試數據
- 測試 / 測試數據
- 測試數據 "Visa" "Amex"
- 測試數據

測試

- 測試數據
- 測試數據 / SMS / 測試數據
- 測試數據
- 測試數據
- 測試數據
- 測試數據 £0.01 測試

測試

- 測試數據
- 測試 60/30/7 測試數據
- 測試數據 / 測試數據
- 增加 Mailjet 測試數據
- 測試數據
- 測試數據

測試

- 銀行の口座番号と銀行名
- 銀行の住所と支店名
- 銀行の電話番号とFAX番号
- 銀行の営業時間と休日
- 銀行のサービスと手数料
- 銀行のセキュリティとプライバシー

銀行

- **銀行** - 銀行の概要
- `payments_process` - 支払い処理
- `payments_invoices` - 支払い請求書
- `features_topup_recharge` - 入金・チャージ機能
- `basics_payment` - 基本的な支払い
- `customer_care` - カスタマーサポート

□ □ □ □

□ □

□□□□□□□□□□□□□□□□“□□”□□

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ " ☐☐ "

□ □

□ □

[illegible]

□ □ □ □ □ □

- [illegible]

Page 10

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

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

□ □ □ □ □ □ □

- □□□□□□□□
- □□□□□
- □□□□□□□□



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

1

-
-
- Stripe
-

📁 basics_payment

📄 2📄

📄 📄📄📄📄📄📄📄

📄📄📄

- 📄📄📄📄📄📄📄📄📄📄📄
- 📄📄📄📄📄📄/📄📄📄📄
- 📄📄📄📄📄📄📄
- 📄📄📄📄📄
- 📄📄📄📄📄📄📄📄

📁 payments_transaction

📄 3📄

📄 📄📄📄📄📄📄📄

📄📄📄

- 📄📄📄📄📄📄📄📄📄📄📄
- 📄📄📄📄📄📄/📄📄📄📄
- 📄📄📄📄📄📄
- 📄📄 PDF
- 📄📄📄📄📄📄📄📄
- 📄📄📄📄Stripe 📄📄📄📄📄

2. 準備

必要なものを準備

- 1 つのアカウント
- 開発環境 (Python 3.7+)
- 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

必要なものを準備

- 環境変数 `Stripe` を設定
- 開発環境 (Python 3.7+)
- 必要なライブラリ

2. 実行

必要なものを Stripe

- 環境変数
- 実行コマンド `#1234`
- 出力結果 `-$45.00`
- 環境変数 `invoice_id` を設定

完了

3. 準備

- 準備するもの
- 準備する手順

準備 → 実行

Stripe 準備

1. Stripe アカウントを作成
2. Stripe の API キーを取得
3. 決済モードを設定
4. Stripe のテストモードに切り替え
5. 決済モードを設定
6. `payment_reference` を Stripe の ID とする

決済モード

1. 決済モード/POS/決済モードを設定
2. 決済モードを設定
3. 決済モードを設定
4. 決済モードを設定
5. 決済モードを設定

決済モード

決済モード

決済モード 1 決済モード

決済モード

1. 決済モード
 - 決済モード → 決済モード
 - 決済モード → 決済モード “決済モード”
 - 決済モード Stripe 決済モード

- 顧客の住所を登録

2. 商品選択

- 商品を選択
 - 商品A \$50.00
 - 商品B \$45.00
 - 商品C \$95.00

3. 注文確認

- 商品名 → 商品 “商品A”
- 数量を選択
- 商品 “商品A”
- 商品 #INV-2025-001234 商品A \$95.00

4. 支払い方法

- 商品番号 #INV-2025-001234
- 商品A 商品B \$0.00

5. 注文完了

- 商品番号 #INV-2025-001234
- 商品A PDF 商品B 商品C

6. 商品詳細

- 商品番号 #INV-2025-001234
- 商品A 商品B
- 商品 “商品A” 商品B
- 商品番号 #INV-2025-001234
- 商品 “商品A”
- Stripe 商品B

7. 商品一覧

- 商品番号 “商品A”
- 商品A 商品B \$95.00

商品 商品番号 #INV-2025-001234

商品 2商品一覧

商品 商品番号 #INV-2025-001234

1. 商品一覧

- 000000103100
- 00000000000000000000
- 00 000000000000

2. 0000000000

- 000 00 00
- 000000000000
- 00000000000000
- 000000000000

3. 000000

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

4. 000000

- 000000000000
- 000000 000000 \$0.00

5. 000000000000

- 0000000000000000
- 000000000000

6. 000000

- 0000000000000000000000
- 000000000000000000/POS 00
- 000000 000000000000

000 00000010000000000000000000

00000 **3**000000000000

000 0000000000000000000000

1. 00000000

- [illegible]

- Stripe 環境構築“準備”

- 環境構築環境構築

2. 環境構築

- 環境構築 環境構築
- 環境構築 環境 → 環境 “環境構築”
- 環境構築環境構築環境構築
- 環境構築

3. 環境構築

- 環境構築環境 “環境構築”
- 環境構築環境構築

4. 環境構築

- 環境構築 環境構築
- 環境構築環境構築
- 環境構築 “環境構築” 環境構築
- 環境構築環境構築環境構築
- 環境構築 “環境構築”
- Stripe 環境構築環境構築

5. 環境構築

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

環境構築 環境構築環境構築環境構築環境構築

環境構築 5環境構築環境構築環境構築環境構築

環境構築 環境構築環境構築環境構築環境構築 - 環境構築環境構築

1. 環境構築

- 環境構築 #INV-2025-001234 環境構築環境構築
- 環境構築環境構築環境構築
- 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 環境構築
- 環境構築環境構築
- 環境構築環境構築 (環境構築)

3. □□□□□

4. □□□□□□□□

5. □□□□□□□□

- [illegible]

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

1.

2.

3. □□□□□□□□

- 0000 #1234 00000
- 000000
- 00 “00” 0000
- 000000“0000 2025-02-10 - 00 #001”

- 姓名 “张明”
- 身份证号 #1234 姓名 “张明”

4. 身份证号

- 身份证号 #1235 姓名张明
 - 姓名“张明” 2025-02-10 - 身份证号 #001”
- 身份证号 #1236 姓名张明
 - 姓名“张明” 2025-02-10 - 身份证号 #001”

5. 身份证号

- 姓名 张明
- 身份证号 123456789012345678
- 身份证号 123456789012345678

6. 身份证号

- 姓名 张明 → 身份证号 123456789012345678 \$300.00
- 身份证号 123456789012345678 “张明” 姓名

姓名 身份证号 身份证号 身份证号 身份证号 身份证号 身份证号

身份证号

身份证号

身份证号

- 身份证号 123456789012345678
- 身份证号 123456789012345678
- 身份证号 123456789012345678
- 身份证号 123456789012345678

身份证号

- 身份证号 123456789012345678 100
- 身份证号 123456789012345678
- 身份证号 123456789012345678
- 身份证号 123456789012345678
- 身份证号 123456789012345678 PDF

📌 目標

- 顧客満足度を向上させる
- 新規顧客の獲得率を20%向上させる
- 既存顧客の再購入率を15%向上させる
- 毎月 Stripe を利用する顧客数を1000人にする

📌 戦略

- 顧客セグメンテーション
- 個人化されたマーケティング
- 定期的な顧客コミュニケーション
- 顧客フィードバックの収集

📌 実行計画

📌 1. 顧客セグメンテーション

- 顧客属性（年齢、性別、職業）によるセグメンテーション
- 購買履歴によるセグメンテーション
- 顧客エンゲージメントによるセグメンテーション

📌 2. 個人化されたマーケティング

- 顧客の興味に基づいた商品推薦
- 顧客の誕生日や記念日に合わせた特別オファー
- 顧客が PDF をダウンロードした後に追加情報提供
- 顧客の過去の購入品に基づいた関連商品の提案

📌 3. 定期的な顧客コミュニケーション

- 定期的なニュースレター配信
- 顧客の質問やフィードバックに対する迅速な対応
- 顧客の成功事例やお客様の声の共有

プラットフォーム

メール

- Mailjet を利用してメールを送信
- メールを PDF に変換
- Stripe と PayPal を利用して決済
- 決済履歴を保存

決済

- Stripe を利用して決済
- Stripe を利用して決済 - 決済履歴を保存
- Stripe を利用して決済 - 決済履歴を保存
- Stripe を利用して決済 - 決済履歴を保存

決済

- Stripe を利用して決済
- Stripe を利用して決済
- Stripe を利用して決済

決済履歴

決済履歴

決済

- Stripe を利用して決済
- Stripe を利用して決済

決済

1. Stripe を利用して決済
2. Stripe を利用して決済
3. Stripe を利用して決済

📋 目錄

- 關於本報告的目的和範圍
- 報告的結構和內容
- 如何閱讀本報告
- 聯繫我們

📌 重要信息

📌 注意

- 本報告僅供參考
- 本報告不構成任何投資建議
- 本報告不承擔任何法律責任

📌 警告

1. 本報告僅供參考
2. 本報告不構成任何投資建議
3. 本報告不承擔任何法律責任
4. 本報告不承擔任何法律責任

📋 目錄

- 關於本報告的目的和範圍
- 報告的結構和內容
- 如何閱讀本報告
- 聯繫我們

📌 重要信息

📌 注意

- 本報告僅供參考
- 本報告不構成任何投資建議
- 本報告不承擔任何法律責任

📌 警告

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

111

- □□□□□□□□ → **B** □□□□□□□□

111

- ১০০০ জনের মধ্যে “১০০” প্রায় ১০ জনের মধ্যে
- ১০০ জনের মধ্যে “১০০০” প্রায় ১ জনের মধ্যে

5/5

- 0000000000000000/000000000000
- 00000000 ID 00000000

□□□□□

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

--	--	--	--

- `basics_payment` - 基礎的支払
- `payment_system_guide` - 決済 API のガイド
- `payments_transaction` - 決済取引
- `payments_invoices` - 決済請求書
- `integrations_mailjet` - Mailjet 連携
- `csa_activity_log` - CSA 活動ログ
- `ocean_rate` - OCS/CGRateS 船運料率

GAAP 零售成本

背景

`customer_transaction` 表 CRM 表 / GL 表 `wallet_ledger` 表 `customer_transaction` 表 `retail_cost` 表 `payment_type` 表

- `wallet_topup_card`: 充值卡 (POST /crm/wallet/topup)
- `wallet_topup_cash`: 现金充值 (POST /crm/wallet/credit)
- `wallet_adjustment_debit`: 充值卡调整 (POST /crm/wallet/debit)

`wallet_ledger_id` 表 `wallet_ledger` 表

GAAP

`retail_cost` 表 CRM 表 `retail_cost` 表 / `retail_cost` 表 CRM 表

GAAP 零售成本 `customer_transaction.retail_cost` 表 `retail_cost` 表 `Cr` / `Dr` 表 / `retail_cost` 表 `Dr` 表

零售成本

零售成本 `POST /crm/payments/invoice` 表 `routing_mode: wallet_only` 表

OmniCRM

OmniCRM 是一個集成的客戶關係管理系統，旨在簡化您的業務流程並提高客戶滿意度。它提供了一個全面的平台，用於管理您的客戶數據、銷售管道、市場營銷活動和客戶服務。OmniCRM 還具有強大的分析功能，可幫助您了解客戶行為並做出明智的決策。

OmniCRM 的主要特點包括：

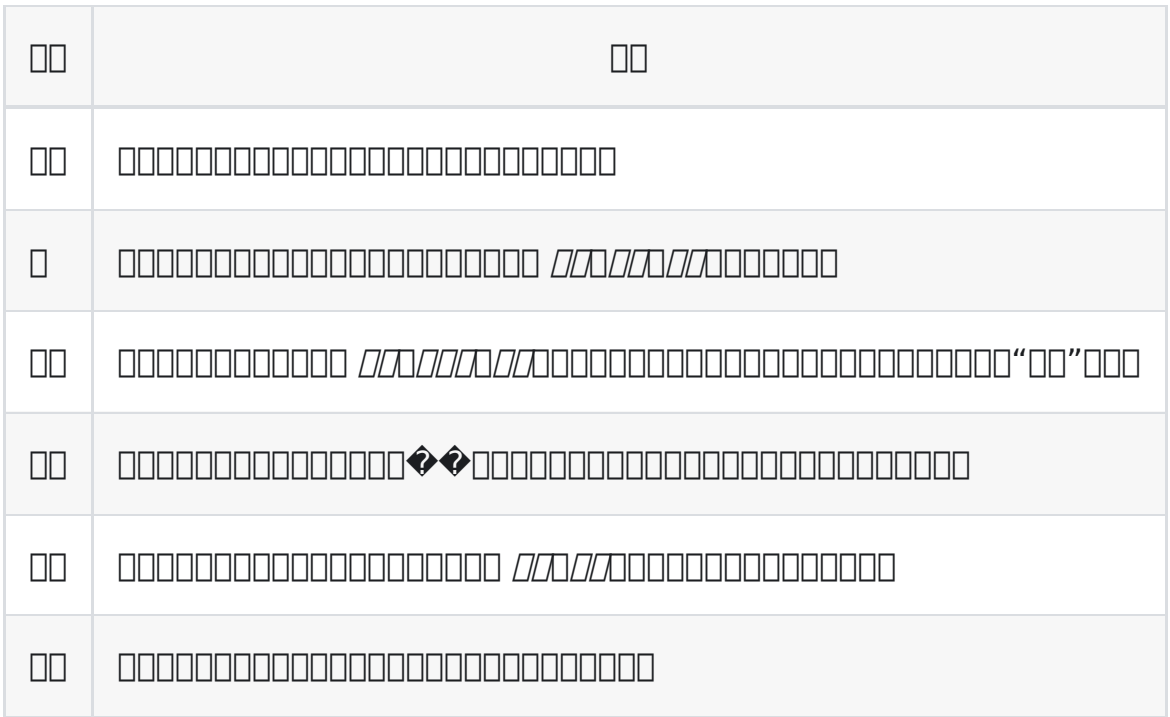
特點

OmniCRM 提供以下功能：

1. 客戶數據管理 (CRM)：集中管理客戶信息，包括聯繫方式、歷史記錄和偏好。
2. 銷售管道管理：跟踪銷售流程，從潛在客戶到成交，確保每個階段都得到適當的關注。
3. 市場營銷自動化：創建和管理營銷活動，如電子郵件系列和社交媒體廣告，以提高客戶參與度。
4. 客戶服務工具：提供多渠道客戶支持，包括電子郵件、電話和聊天，以確保客戶問題得到快速解決。
5. 報告和儀表板：實時監控業務性能，生成各種報告和儀表板，以便您了解業務趨勢。
6. 集成和兼容性：與各種第三方應用程序集成，如電子郵件客戶端、會計軟件等，以實現數據同步。
7. 安全性和隱私保護：採用嚴格的加密和訪問控制措施，確保客戶數據的安全性和隱私。

安裝

OmniCRM 的安裝過程非常簡單，只需按照以下步驟操作即可：

[illegible]

{PREFIX} - {BOARD} - {SEQUENCE}

BRD-001-00001

PREFIX	BRD 10 INST
BOARD	ID
SEQUENCE	

API

-
-
- -
 -
 - BRD 10
 - API
-

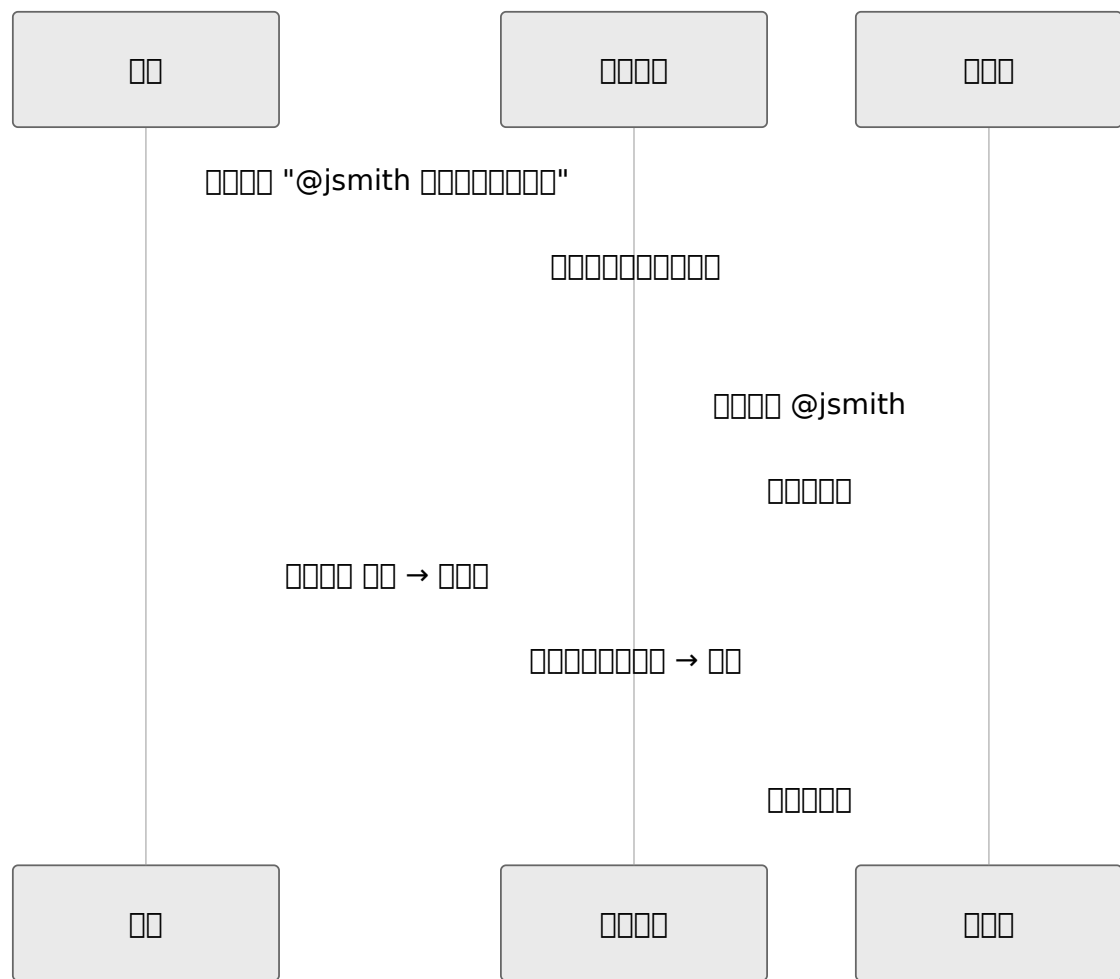
項目	内容
名称	XXXXXXXXXXXX
住所	XXXXXXXXXX
性別	XXXXXXXXXXXXXXXXXXXXXXXXXXXX
年齢	XXXXXXXXXXXXXXXXXXXXXXXXXXXX
職業	XXXXXXXXXXXXXXXXXXXXXXXXXXXX
電話番号	XXXXXXXXXX XXXX
メールアドレス	XXXXXXXXXX
婚姻状況	XXXXXXXXXX
収入状況	XXXXXXXXXXXXXXXXXXXXXXXXXXXX
家族構成	XXXXXXXXXX / XXXX / XXXX
その他	XXXXXXXXXXXXXXXXXXXX

お問い合わせ

お問い合わせ先    XXXXXXXXXX

- XXXXXXXXXXXXXXXXXXXX/XXXX
- XXXXXXXXXXXXXXXXXXXX
- XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX



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

[illegible]

- 00000000000000000000000000000000
- @00000000000000 @username 00000000
- 00000000000000000000000000000000

□□□□□□□□□□□□□□□□

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

[illegible]

❓❓❓

API	API Description
VIEW_BOARD	View Board
CREATE_BOARD	Create Board
UPDATE_BOARD	Update Board
DELETE_BOARD	Delete Board
MANAGE_BOARD_SETTINGS	Manage Board Settings
MANAGE_BOARD_MEMBERS	Manage Board Members
GENERATE_PUBLIC_TOKEN	Generate Public Token
VIEW_BOARD_ITEM	View Board Item
CREATE_BOARD_ITEM	Create Board Item
UPDATE_BOARD_ITEM	Update Board Item
DELETE_BOARD_ITEM	Delete Board Item

API

API

API	API
API	API
API	API

CGRateS

CGRateS OmniCRM

11

☐ OmniCRM ☐ (CGRateS) ☐

CGRateS Playbooks

□ □ □ □

CDR

-

ID -

□□ - □□□□□□□□□□□□□□□□

[illegible][illegible]

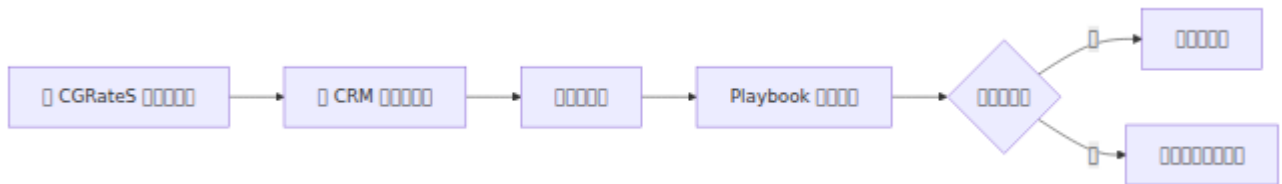
```

00000000 ExecuteAction 0 Playbook 0000000000000000 CGRateS 000000000000
00000000

```

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

Python による CRM の OCS



CRM (blue box)

CGRateS (blue box)

- **CGRateS** (blue box) `ActionsId = "Action_50gb-data-pack"`
- **CRM** (blue box) `product_slug = "50gb-data-pack"`
- **Playbook** (blue box) `cgr_action_name = "Action_" + product_slug`

Playbook (blue box) `product_slug` (blue box) `ExecuteAction` (blue box) `CGRateS` (blue box)

CGRateS (blue box)

CRM (blue box)

1. CGRateS (blue box) - CRM (blue box)
2. CRM (blue box) - `product_slug` (blue box)
3. CRM (blue box) - `OCS` (blue box) `CRM` (blue box) `Python` (blue box)

CGRateS (blue box)

```

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)

```

XXXXXXXXXXXX

```

# 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 格式

```
# 数据对象
{
  "id": "Data_5GB_5days_uuid_abc123",
  "data": {
    "size": 5368709120, # 5GB 大小
    "date": "2024-12-29",
    "count": 10
  }
}

# 数据对象
{
  "id": "Data_10GB_30days_uuid_def456",
  "data": {
    "size": 10737418240, # 10GB 大小
    "date": "2025-01-24",
    "count": 10
  }
}

# 数据对象
# 数据对象
```

CGRateS 数据对象 - 数据对象

*topup *topup_reset

ID 数据对象

*topup - 数据对象

*topup 数据对象 ID 数据对象

数据对象

- 数据对象 ID 数据对象
- 数据对象
- 数据对象
- 数据对象

000

```
# 00000
000
ID: "Data_Package__5368709120"
0: 1073741824 # 00 1GB
00: 2024-12-24 (00 1 0)

# 00 *topup 000
000
000: "*topup"
000
ID: "Data_Package__5368709120" # 00 ID - 0000
0: 5368709120 # 5GB
ExpiryTime: "+5d"

# *topup 00000
000
ID: "Data_Package__5368709120"
0: 6442450944 # 6GB (1GB + 5GB 00)
OriginalValue: 5368709120 # 000000 5GB
Value_hr: "6 GB"
OriginalValue_hr: "5 GB"
Remaining_hr: "6 GB of 5 GB (1 GB 00)"
00: 2024-12-29 (0000 5 0)
```

000

- 00000000000000000000
- 0000000000000000
- 00000
- 00000

*topup_reset - 0000

*topup_reset 00 0000000000000000 ID0

000

- 000000 ID 00000

- 0000000000
- 0000000000
- 0000000000

000

```
# 000000
000
ID: "Data_Package__5368709120"
0: 1073741824 # 00 1GB
00: 2024-12-24 (00 1 0)

# 00 *topup_reset 000
000
000: "*topup_reset"
000
ID: "Data_Package__5368709120" # 00 ID - 0000
0: 5368709120 # 5GB
ExpiryTime: "+5d"

# *topup_reset 000000
000
ID: "Data_Package__5368709120"
0: 5368709120 # 5GB (00 1GB 000)
OriginalValue: 5368709120
Value_hr: "5 GB"
Remaining_hr: "5 GB of 5 GB"
PercentUsed: 0
00: 2024-12-29 (0000 5 0)
```

000

- 000000000000000000
- 000000000000000000
- 000000000000000000
- 000000000000000000

OmniCRM ID 取得手順

OmniCRM ID 取得手順

OmniCRM ID 取得手順

OmniCRM には、API 経由で ID 取得を行うことができます。API 経由で ID 取得を行う場合は、Web UI 経由で ID 取得を行う必要はありません。

OmniCRM ID 取得

```
{DescriptiveName}__{OriginalSizeInBaseUnits}
```

OmniCRM ID

```
# 取得するID
"AU_Data_Domestic__107374182400"

# 取得するID
# - 取得するID "AU_Data_Domestic" 取得するID - 取得するID
# - 取得するID "107374182400" 取得するID
# - UI 取得するID "AU Data Domestic - 100 GB"

# 取得するID
"AU_Voice_Domestic__1800000000000000"
# - 取得するID "AU_Voice_Domestic" 取得するID "AU_Voice_Domestic_3000min"
# - 取得するID "1800000000000000" 取得するID
# - UI 取得するID "AU Voice Domestic - 3000 min"

# 取得するID
"AU_SMS_Domestic__3000"
# - 取得するID "AU_SMS_Domestic"
# - 取得するID "3000" 取得するID
# - UI 取得するID "AU SMS Domestic - 3000 msgs"
```

OmniCRM ID 取得手順 ☐ API 経由で ID 取得を行う

API 経由で ID 取得

OmniCRM API CGRateS ID _hr

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

        // 00
        "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": "20250125002200"
      }
    ]
  }
}
```

API

1. ID

```
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.

```
# 国内データ
id_hr = descriptive_name.replace("_", " ") # "AU Data Domestic"
```

3. 単位変換

```
# 国内データ
if balance_type == "*data":
    original_value_hr = convert_bytes_to_gb(original_size) #
    "100 GB"

# 国内データ
elif balance_type == "*voice":
    original_value_hr = convert_ns_to_minutes(original_size) #
    "3000 min"

# 国内データ
elif balance_type == "*sms":
    original_value_hr = f"{original_size} msgs" # "3000 msgs"
```

4. 使用率の計算

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

5. 残り容量の表示

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

Web UI 構築

残り容量 国内データ

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

// 表示範囲
<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 範囲 "50 GB of 100 GB" 範囲 "50 GB 日"
2. 範囲 - 範囲範囲範囲範囲範囲範囲
3. 範囲 - 範囲 ID 範囲範囲範囲範囲範囲範囲範囲
4. 範囲 - 範囲 ***topup**範囲範囲範囲範囲 70 GB 範囲 100 GB
 - 範囲 ID 範囲範囲"AU_Data_Domestic__107374182400"範囲 100 GB
 - 範囲範囲170 GB
 - UI 範囲"170 GB (70 GB 範囲 + 100 GB 範囲)"

範囲 - 範囲 **ID**

範囲  範囲範囲範囲範囲範囲 UI 範囲範囲範囲範囲範囲範囲

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

```
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
    }
  ]
}
```

- 00000000UI 00000000

```
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 を利用する。

UUID

```

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

- name: 通过 *topup 充值套餐
  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
          }
        }]
      }

```

通过 通过 *topup 充值套餐


```
# 5days 5GB of 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

# 15GB
# UI "Data 5days - 5 GB of 5 GB"
```

2 *topup ID

ID *topup

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

- name: 充值 *topup 充值
  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
          }
        }]
      }
}
```

□□□ □□□ ◆◆ □□□□□□□□□□□□□□□□

1 5GB

Info

ID: "Data_5days__5368709120"

ID: 5368709120

Value_hr: "5 GB"

Remaining_hr: "5 GB of 5 GB"

PercentUsed: 0

Expire: 2024-12-29

3 1GB

Info

ID: "Data_5days__5368709120"

ID: 9663676416 # 4GB + 5GB

Value_hr: "9 GB"

OriginalValue_hr: "5 GB"

Remaining_hr: "9 GB (4 GB + 5 GB)"

PercentUsed: -80 #

Expire: 2024-12-27 5

3 *topup_reset ID

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 * 0.9;
CGRateS = CGRateS * 0.9;
```

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

Blocker: true

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

準備

- 事前に 0CS 側の設定が完了していること
- 0CS 側の設定項目は CGRateS の 0CS 設定
- 0CS 側の設定項目は *voice *data *sms *monetary

0CS 側の設定

1. 0CS 側の設定

0CS 側の設定項目は CGRateS の 0CS 設定

```
Action_Suspend_Account = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_suspend-account",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # 0CS 側の設定項目
      {
        "Identifier": "*topup",
        "BalanceId": "Suspension_Blocker",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any",
        "Units": 0, # 0
        "BalanceWeight": 9999, # 0.0001 - 0.0001
        "Blocker": True, # 0.0001
        "Weight": 10
      }
    ]
  }]
}

result = OCS_Obj.SendData(Action_Suspend_Account)
```

0CS 側の設定項目は CGRateS の 0CS 設定

2. 0CS 側の設定

0000000000000000

```
Action_Monthly_Plan_With_Cap = {
  "id": "0",
  "method": "ApierV1.SetActions",
  "params": [{
    "ActionsId": "Action_monthly-with-cap",
    "Overwrite": True,
    "Tenant": tenant,
    "Actions": [
      # 10GB
      {
        "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
      }
    ]
  }
}]
}
```

000

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

3. Overage_Cap\$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": "Dest_Domestic_All",
        "Units": 10000, # $100
        "ExpiryTime": "*month",
        "BalanceWeight": 1000,
        "Weight": 20
      },
      # 充值卡充值0900
      {
        "Identifier": "*topup",
        "BalanceId": "Premium_Blocker",
        "BalanceType": "*monetary",
        "DestinationIDs": "Dest_Domestic_Premium",
        "Units": 0, # 0
        "BalanceWeight": 2000, # 充值卡 - 普通充值
        "Blocker": True,
        "Weight": 10
      }
    ]
  }]
}

```

说明

- 普通充值 → 普通Regular_Usage
- 充值卡充值 → 充值Premium_Blocker 2000 > 1000 → 充值

□□□□□□□□

□□□

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
      {
        "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
      {
        "Identifier": "*topup_reset",
        "BalanceId": "Overage_Allowance",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any",
        "Units": 2000, # $20.00
        "ExpiryTime": "*month",
        "BalanceWeight": 1000,
        "Weight": 90
      },
      # $50
      {
        "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 00Domestic_Voice_500min0000
2. **500-700**000 00Overage_Allowance\$0.10/00 = \$200200000
3. **700-1200**000 00Hard_Spending_Cap\$0.10/00 = \$500500000
4. 0**1200**0000 Hard_Spending_Cap00 → 00000000

000000120000000000\$50000000

000000000000

1. 0000000000

```

"BalanceWeight": 9999 # 0000000000

```

2. 000000000000

```

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

```

3. 000000000000

- 00ActionTriggers0000000080%090%0100%000000
- 00000000000000000000

4. 000000000000

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

5.

- `INSUFFICIENT_CREDIT_BALANCE_BLOCKER`
- `CDR` = -1.0
-

- 1.
2. `DestinationIDs`
3. `True`

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

# 
# - 
# - BalanceWeight
# - DestinationIDs
# - = 0
```

Blocker: true

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

1

[illegible]

#

11

DestinationIDs:

1

00

11

DestinationIDs:

1

11

#

#

[illegible]

2

□ □

```

[[1]]
  ID: "Premium_Data"
  : 5GB
  : 20

[[2]]
  ID: "Standard_Data"
  : 10GB
  : 10

# [[1]] > [[2]]
# [[2]]

```

[[1]]

[[3]]

[[1]]

```

[[1]]
  ID: "Data_Package_A"
  : 5GB
  : 2024-12-25
  : 10

[[2]]
  ID: "Data_Package_B"
  : 10GB
  : 2025-01-15
  : 10

```

[[1]]

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

包2
  ID: "Data_Package_B"
  大小: 10GB
  日期: 2025-01-15
  索引: 10 # 索引 = 偏移量

# 包1的索引11 > 包10
```

数据包 数据包的元数据 数据包的元数据 数据包的元数据

数据

包1数据包的元数据

包 数据包的元数据“5GB 5”数据包的元数据

包


```

- name: 随机生成UUID
  set_fact:
    uuid: "{{ 99999999 | random | to_uuid }}"

- name: 随机生成5GB
  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
            "Weight": 10
          }
        }]
      }

```

测试

- 12月24日测试 → 5GB 12月29日
- 12月25日测试 → 5GB 12月30日
- 测试数据生成时间间隔10分钟

2. 测试

测试 “5GB” 数据生成时间间隔

测试

```
- name: 0000000000
  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
            "Weight": 10
          }
        }]
      }
}
```

□□□□ □□□□ *topup □□□□□□□□

□□□□□

- 1000 50GB 3000
- 2000 30GB 20GB
- 2000 → 70GB 20GB + 50GB 2000 3000

□□**3**□□□□□□□□□**?****?****?**□□

“100GB”

111

```
- name: 设置动作配置
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: 设置动作计划配置
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: 设置动作计划配置
```

```
uri:
  url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
  method: POST
  body_format: json
  body:
    {
      "method": "ApierV2.SetAccount",
      "params": [{
        "Account": "{{ service_uuid }}",
        "ActionPlanIds": ["ActionPlan_Monthly_100GB"],
        "ReloadScheduler": true
      }]
    }
  }
```

□□□□□

- □1□□□□□100GB□□□95GB□□□5GB
- □2□□□□□□□□100GB□5GB□□□□□□□□
- □2□□□□□20GB□□□80GB
- □3□□□□□□□□100GB□80GB□□□□□□□□

□□**4**□□□□□□□□□□

□□□ □□□□“□□□□”□□□□□□□□“□□□□”□□□□□□□□□□□□□□□□

□□□

```
# 测试添加余额
- name: 测试添加余额
  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天
            "Weight": 20 # 20%
          }
        }]
      }
}
```

```
# 测试添加余额
- name: 测试添加余额
  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天
            "Weight": 10 # 10%
          }
        }]
      }
}
```

□□□□

- □□5GB□□□□□□20□+ 50GB□□□□□□10□
- □□□□□□5GB□□□□
- □□□50GB□□□□□□□□
- □□□□□□□□□□

□□□□□□□□

CGRateS□□□□□□□□□□□□□□□□

□□□□

***topup** - □□□□□□□□□□ ***topup_reset** - □□□□□□□□□□□□ ***debit** - □□□□□□
***debit_reset** - □□□□□□□□ ***reset_account** - □□□□□□

□□□□□□□□

□OmniCRM□□□□□□□□□□□□□□□□

□□**1**□□□□□□□□□□□□

□□□□□ → □□□□□ID□UID□ □□□□□ → □□□□□□ID□***topup_reset** □□□□□ → □□□
□□□ID□***topup**

□□**2**□□□□□□□□□□□□□□□□

□□ → □□***topup**□□ □□ → □□***topup_reset**□□ □□□ → □□□□□□ID

□□**3**□□□□□□□□□□□□

□□□□ → □□□□□□□□□□□□/□□□□□□□□□□□□□□□□ □□□□ → □□□□□□□□□□□□ = □□□□□□□□ □□□□ →
□□□□□□30□□□□□□20□□□□□□10□□□□□ □□/□□□□ → □□□□□□□□□□□□□□□□

4

→ +720h 30 → ActionPlan *monthly →
*unlimited

CGRateSPlaybooks

```

- 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 - 平衡タイプ

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

Units - 単位

ExpiryTime - 有効期限

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

BalanceId - 残高ID = 残高ID = 残高

Weight - 重さ = 重さ

Blocker - true/false

Disabled - true/false

Python

Python `cgrateshttpapi` `OCS_Obj.SendData()`

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()))
```

準備

準備するデータ

準備するデータ

- 国コード - 国/SMS国コードを指定する `Dest_International_UK`
- **PLMN** - 国/SMS国コードを指定する `Dest_PLMN_OnNet` `Dest_PLMN_US_Verizon`

準備するデータ

- 国/SMS国 → 国/SMS国コードを指定する
- 国/SMS国 → 国/SMS国コードを指定する

準備するデータ

- 国/SMS国コードを指定する
- PLMN国コードを指定する
- PLMN国コードを指定する
- 国/SMS国コードを指定する

国/SMS国コードを指定する `CGRateS` 国/SMS国コード

準備

準備するデータ

```
# 内存大小
1_GB = 1 * 1024 * 1024 * 1024 # 1073741824 字节
100_GB = 100 * 1024 * 1024 * 1024

# 时间大小
1_minute = 60 * 1000000000 # 60 秒
3000_minutes = 3000 * 60 * 1000000000

# SMS大小
3000_sms = 3000
```

1 Python

SMS

```

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流量
            # 重置PLMN流量
            # 重置PLMN
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Data_Domestic__" + str(100 * 1024
* 1024 * 1024),
                "BalanceType": "*data",
                "DestinationIDs": "Dest_PLMN_OnNet", # 重置PLMN
                #mcc505.mnc057
                "Units": 100 * 1024 * 1024 * 1024,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200,
                "Weight": 90
            },
            # 3000分钟
            {
                "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 SMS
            {

```

```

        "Identifier": "*topup_reset",
        "BalanceId": "AU_SMS_Domestic__" + str(3000),
        "BalanceType": "*sms",
        "DestinationIDs": "Dest_AU_Mobile;",
        "Units": 3000,
        "ExpiryTime": "*month",
        "BalanceWeight": 1200,
        "Weight": 88
    },
    # 6GB
    {
        "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
    },
    # CDR
    {
        "Identifier": "*cdrlog",
        "BalanceId": "",
        "BalanceUuid": "",
        "BalanceType": "*generic",
        "Directions": "*out",
        "Units": 0,
        "ExpiryTime": "",
        "Filter": "",
        "TimingTags": "",
        "DestinationIds": "",
        "RatingSubject": "",
        "Categories": "",
        "SharedGroups": "",
        "BalanceWeight": 0,
        "ExtraParameters": "
{\"Category\\\": \"^activation\\\", \"Destination\\\": \"AU Premium Plan
1\\\"}\",
        "BalanceBlocker": "false",
        "BalanceDisabled": "false",
        "Weight": 80
    }
}

```

```

    ]
    }]
}

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

```

测试

- *reset_account
- *topup_reset
- BalanceWeight 1200 > 1100
- Weight
- *cdrlog

2 Python

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套餐
            # 套餐名称PLMN标识符
            {
                "Identifier": "*topup_reset",
                "BalanceId": "AU_Data_Domestic__" + str(20 * 1024
* 1024 * 1024),
                "BalanceType": "*data",
                "DestinationIDs": "Dest_PLMN_OnNet", # 套餐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("套餐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
            },
            # 10000000000
            {
                "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")

```

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

[illegible]

Python

□ □ □ □ □ □ □

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

- **Actions** -

- **Identifier** - *topup *topup_reset *reset_account *cdrlog
- **BalanceId** -
- **BalanceType** - *data *voice *sms *monetary
- **DestinationIDs** -
 - /SMS
 - "Dest_AU_Mobile" "Dest_International_UK"
 - PLMN "Dest_PLMN_OnNet" "Dest_PLMN_US_Verizon"
- **Units** - SMS
- **ExpiryTime** - *month +720h 2024-12-31
- **BalanceWeight** -
- **Weight** -
- **Blocker** - True
- **Disabled** - True

```
- name: 
  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"
        }]
      }
}
```

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

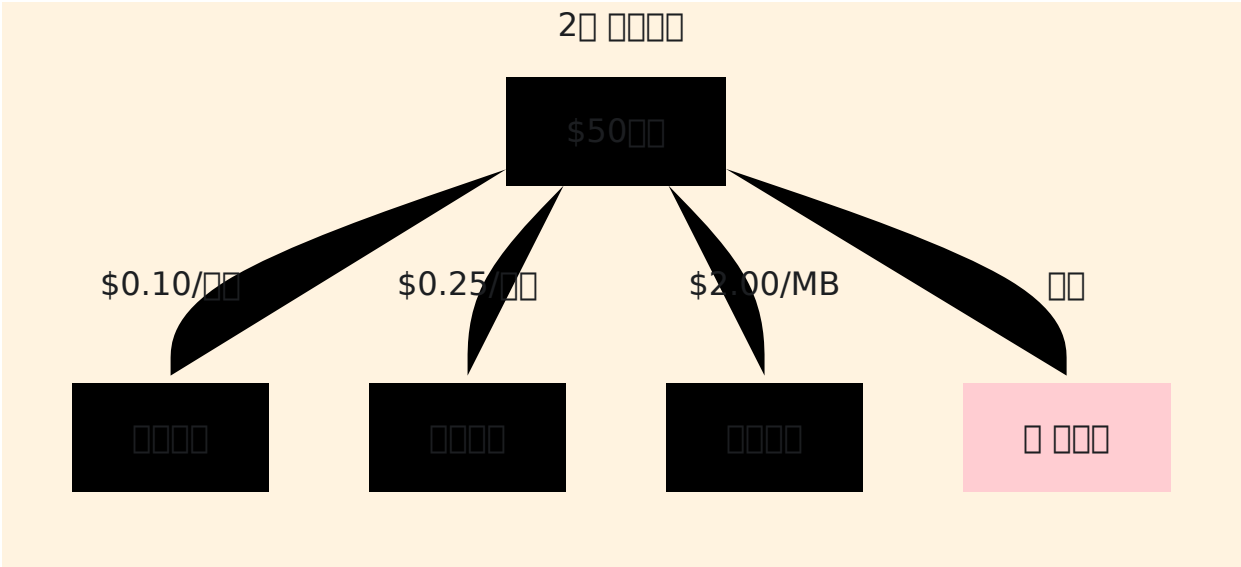
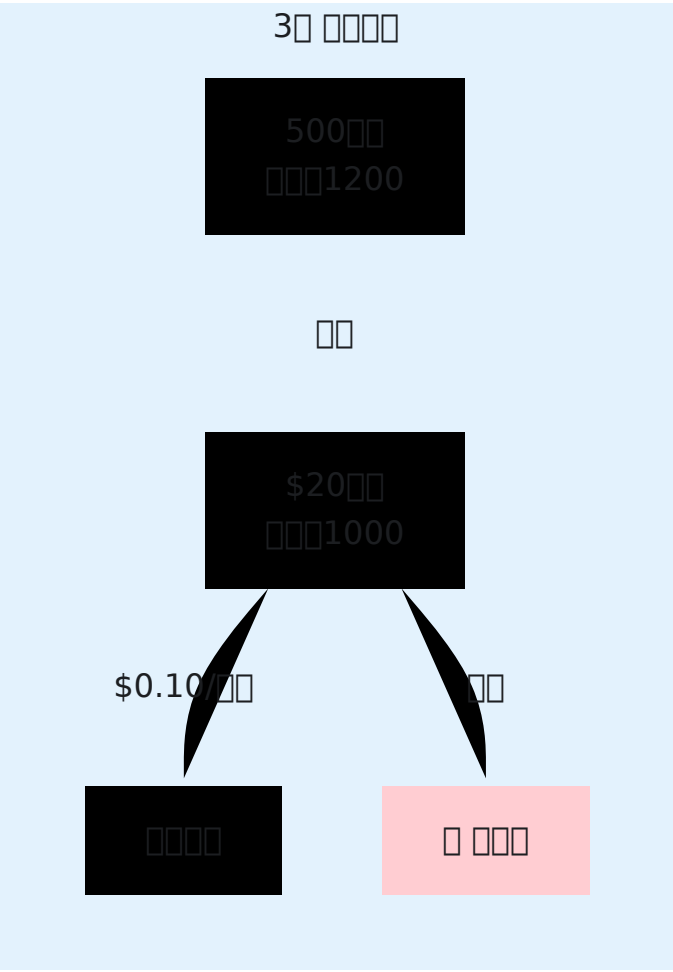
□ □ □ □ □ □

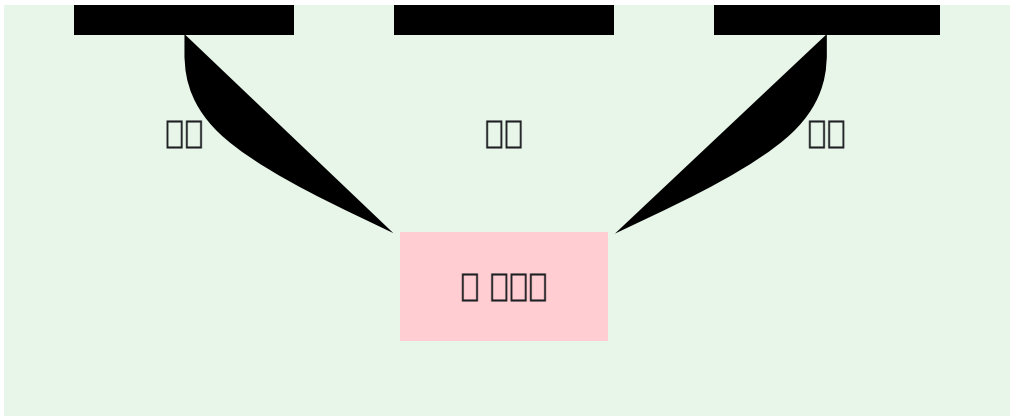
CGRateS

□□□□□□ **VS** □□□□ **VS** □□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

項目	定義	計算方法	単位
総売上高	全商品・サービスの売上合計	各商品の売上を合計	円
総売上原価	全商品・サービスの仕入れコスト合計	各商品の仕入れ額を合計	円
粗利益	総売上高から総売上原価を差し引いた金額	総売上高 - 総売上原価	円
営業費用	営業活動に必要とする経費の総和	各営業費用を合計	円
営業利益	粗利益から営業費用を差し引いた金額	粗利益 - 営業費用	円
経常利益	営業利益とその他の経常的な収益・損失を加算した金額	営業利益 + 経常的収益 - 経常的損失	円
税引前利益	経常利益と法人税等の税金を差し引いた金額	経常利益 - 法人税等	円
純利益	税引前利益から法人税等の税金を差し引いた最終的な利益	税引前利益 - 法人税等	円
株主一人当たり純利益	純利益を発行済み株式数で割った金額	純利益 ÷ 発行済み株式数	円/株





1

GB SMS

```

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

```


□□ **2**□□□□□□□□

CGRateS

[illegible]

□□□\$50□□□□□□

```

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)

```

1000000000


```

Action_Hybrid_Plan = {
    "id": "0",
    "method": "ApierV1.SetActions",
    "params": [{
        "ActionsId": "Action_hybrid-flex-plan",
        "Overwrite": True,
        "Tenant": tenant,
        "Actions": [
            {
                "Identifier": "*reset_account",
                "Weight": 700
            },

            # ===== 充值重置 =====
            # 500分钟
            {
                "Identifier": "*topup_reset",
                "BalanceId": f"Domestic_Voice__{{500 * 60 *
1000000000}}",
                "BalanceType": "*voice",
                "DestinationIDs": "Dest_Domestic_All",
                "Units": 500 * 60 * 1000000000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1200, # 充值重置
                "Weight": 95
            },
            # 100分钟
            {
                "Identifier": "*topup_reset",
                "BalanceId": f"International_Voice__{{100 * 60 *
1000000000}}",
                "BalanceType": "*voice",
                "DestinationIDs": "Dest_International_All",
                "Units": 100 * 60 * 1000000000,
                "ExpiryTime": "*month",
                "BalanceWeight": 1150, # 充值重置
                "Weight": 94
            },
            # 1000条SMS
            {
                "Identifier": "*topup_reset",
                "BalanceId": "Domestic_SMS__1000",
                "BalanceType": "*sms",

```

```

        "DestinationIDs": "Dest_Domestic_All",
        "Units": 1000,
        "ExpiryTime": "*month",
        "BalanceWeight": 1200,
        "Weight": 93
    },
    # 15GB套餐
    {
        "Identifier": "*topup_reset",
        "BalanceId": f"Domestic_Data__{{15 * 1024 * 1024 *
1024}}",
        "BalanceType": "*data",
        "DestinationIDs": "Dest_PLMN_OnNet",
        "Units": 15 * 1024 * 1024 * 1024,
        "ExpiryTime": "*month",
        "BalanceWeight": 1200,
        "Weight": 92
    },
    # 2GB套餐1
    {
        "Identifier": "*topup_reset",
        "BalanceId": f"Roaming_Zone1_Data__{{2 * 1024 *
1024 * 1024}}",
        "BalanceType": "*data",
        "DestinationIDs": "Dest_PLMN_Zone_NorthAmerica",
        "Units": 2 * 1024 * 1024 * 1024,
        "ExpiryTime": "*month",
        "BalanceWeight": 1100,
        "Weight": 91
    },

    # ===== 套餐/套餐 =====
    # $20套餐
    {
        "Identifier": "*topup",
        "BalanceId": "PAYG_Overflow_Balance",
        "BalanceType": "*monetary",
        "DestinationIDs": "*any",
        "Units": 2000, # $20.00
        "ExpiryTime": "*month",
        "BalanceWeight": 1000, # 
        "Weight": 90
    },

```

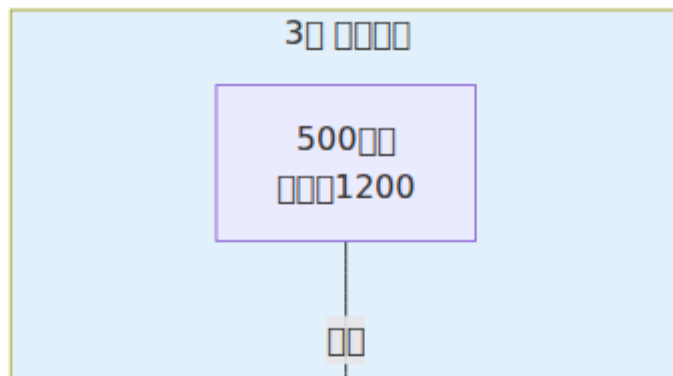
```

        {
            "Identifier": "*cdrlog",
            "BalanceType": "*generic",
            "ExtraParameters": "
{\"Category\\":\"^activation\\",\"Destination\\":\"Hybrid Flex
Plan\\"}",
            "Weight": 80
        }
    ]
}

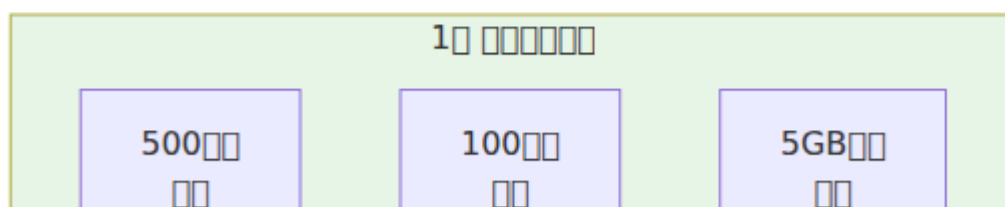
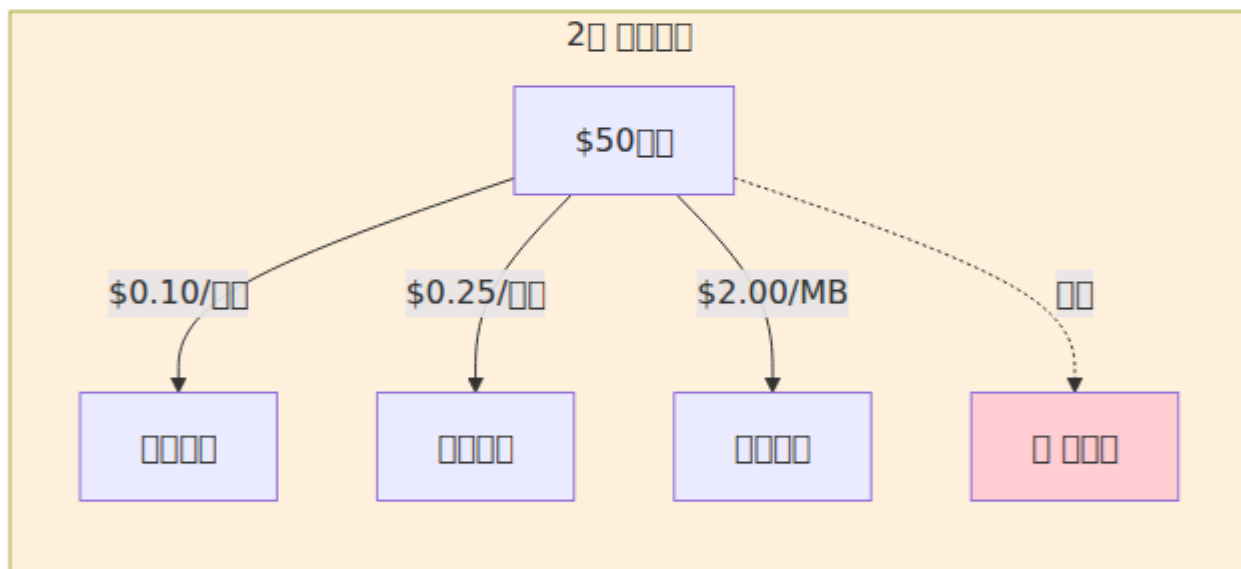
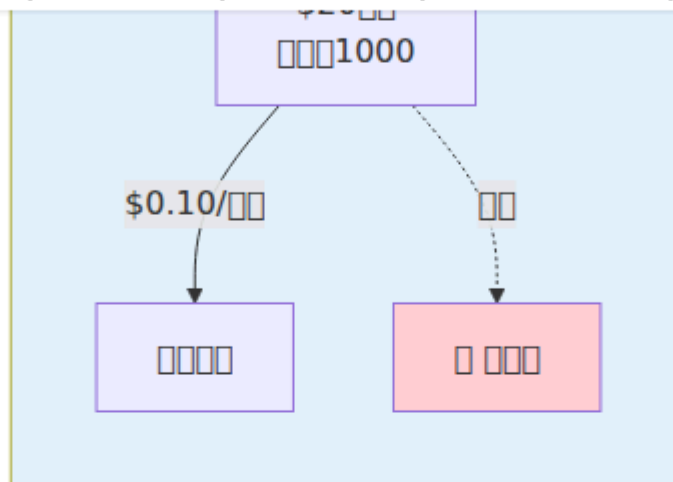
```

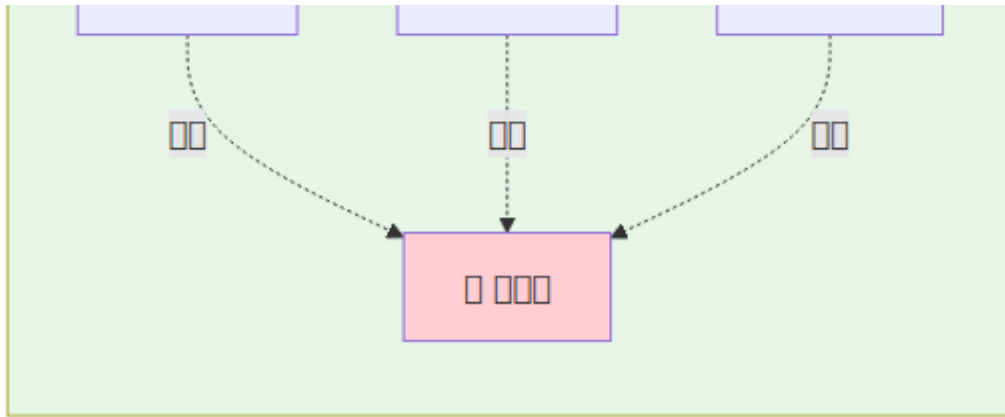
```
result = OCS_Obj.SendData(Action_Hybrid_Plan)
```

□□□□□□□□



ore OmniCore OmniCall OmniRAN OmniCharge Platform 套餐





00000

00**1000600**00000000

1. 50000 → “Domestic_Voice_500min”0001200 - 00
2. 000010000 → “Domestic_Voice_500min”000000“PAYG_Overflow_Balance”00
010000
3. 00010000 × \$0.10 = \$10.00 00000000
4. 000\$10.0000

00**20150**000000000000

1. 10000 → “International_Voice_100min”0001150 - 00
2. 00005000 → 000“PAYG_Overflow_Balance”
3. 000050 × \$0.25 + \$0.0500 = \$12.55
4. 000\$7.450000000000\$200

000

- 0000000000
- 0000000000
- 00000000
- 000000 - 00000

000

- 000000
- 0000000000
- 000000000000

Verizon

Verizon Wireless International Service

1. International Service

Verizon Wireless International Service 500 minutes 1000 texts 10GB

Verizon Wireless International Service 44-20-7946-0958 10

Verizon

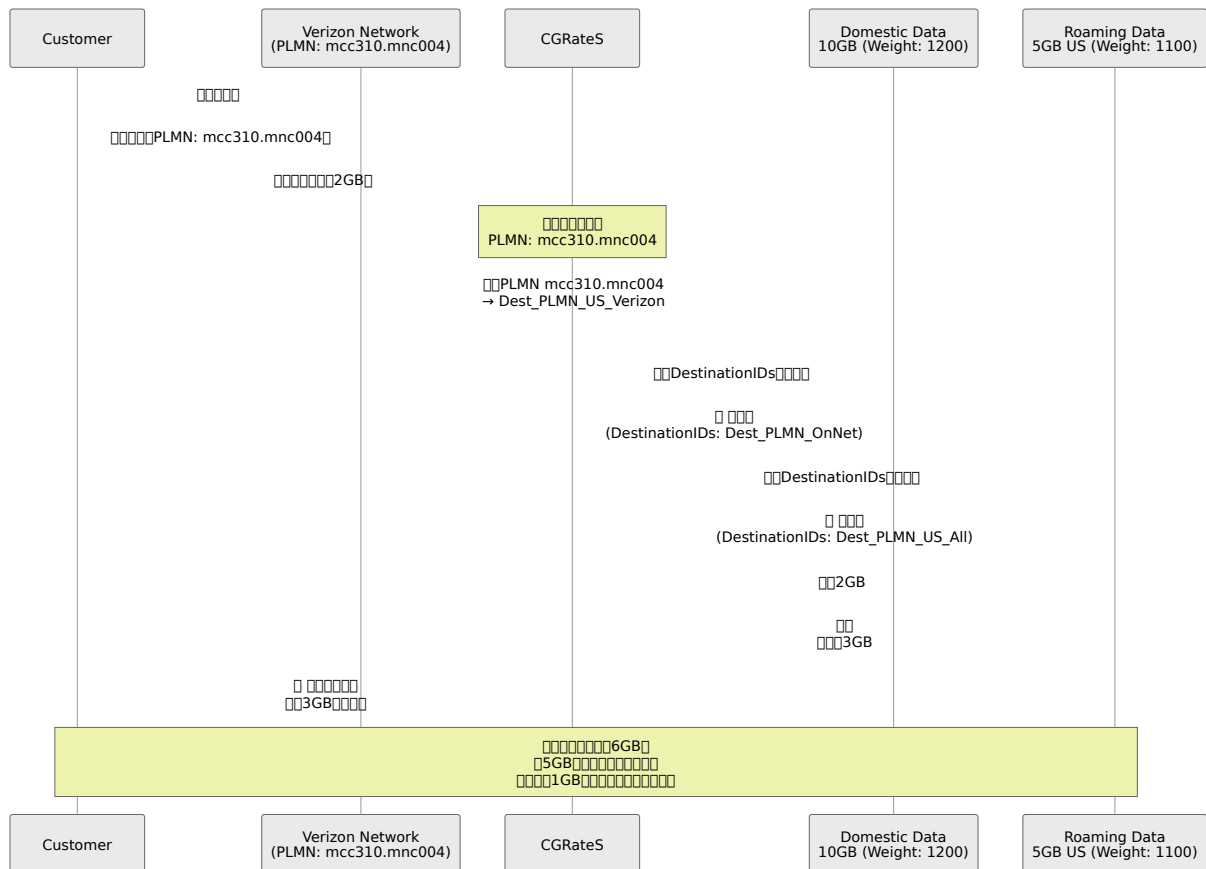
1. CGRateS 44207946...
2. Dest_International_UK
3. DestinationIDs: "Dest_International_UK"
- 4.
- 5.
- 6.
- 7.

Verizon Wireless International Service

2. International Service

Verizon Wireless International Service + 5GB

Verizon Wireless International Service PLMN mcc310.mnc004 2GB



1. CGRateS

1. CGRateS PLMN mcc310.mnc004

2. Dest_PLMN_US_Verizon

3. DestinationIDs: "Dest_PLMN_US_All"

4. "Roaming_US_Data_5GB" BalanceWeight: 1100

5. 2GB

6. 3GB

6GB

1. 5GB →

2. 1GB →

3. → 5GB

3

500 + \$20

600

Example

1. 500 minutes
 - “Domestic_Voice_500min” BalanceWeight: 1200
 - 100
2. “Domestic_Voice_500min”
3. 100 minutes
 - “PAYG_Overflow_Balance” BalanceWeight: 1000
 - \$0.10/minute
 - $100 \times \$0.10 = \10.00
 - \$20
 - \$10.00

Example

- 500 minutes + 100 minutes \$10.00

4. Example

Example \$50

Example Verizon 100MB

Example

1. Verizon 100MB
2. \$2.00/MB
3. $100 \times \$2.00 = \200.00
4. \$50.00
5. 25MB
6. \$50

Example

需求

1. 用户ID

用户ID

```
# 用户ID
"Data_5GB_5days_{{ uuid }}"
"Voice_100min_Monthly"
"Bonus_Data_Loyalty"

# 用户ID
"balance1"
"data"
"temp"
```

2. 用户ID

用户ID

```
# 用户ID
# 30 = 30/10000
# 20 = 20/10000
# 10 = 10/10000
# 5 = 5/10000
```

3. CDR

CDR

```
{
  "Identifier": "*cdrlog",
  "BalanceType": "*generic",
  "ExtraParameters": "
{\"Category\":\"^activation\", \"Destination\":\"{{ package_name
}}\"}"
}
```

4. ActionPlans

Configuration of ActionPlans

```
# Configuration of ActionPlan
- name: ActionPlan
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body_format: json
    body:
      {
        "method": "ApierV1.SetActionPlan",
        "params": [{
          "Id": "ActionPlan_Monthly_Reset",
          "ActionPlan": [{
            "ActionsId": "Action_Monthly_Reset",
            "Time": "*monthly",
            "Weight": 10
          }],
          "ReloadScheduler": true
        }]
      }
}
```

5. ActionTriggers

Configuration of ActionTriggers

```
- name: 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
        }]
      }
}
```

6. 配置

配置

```
# 配置
BalanceWeight: 1200 # 配置
BalanceWeight: 1150 # 配置
BalanceWeight: 1100 # 配置
BalanceWeight: 1000 # 配置
```

配置

7. DestinationIDs配置

配置DestinationIDs配置

```
# 1 - 国際電話
{
  "BalanceType": "*voice",
  "DestinationIDs": "Dest_International_UK", # 英国
  "Units": 200 * 60 * 1000000000
}

# 2 - 国内電話
{
  "BalanceType": "*voice",
  "DestinationIDs": "*any", # 国内任意
  "Units": 200 * 60 * 1000000000
}
```

8. 国際電話

国際電話の料金を設定します。

```
# 1 - 国際電話
"Dest_International_Europe" = 500 # 500円

# 2 - 国内電話
"Dest_France", "Dest_Germany", "Dest_Italy" = 500 # 500円
```

CGRateS Destinations を参照してください。

9. 国際電話

国際電話の料金を設定します。

```
# 1 - 国際電話
{
  "BalanceType": "*monetary",
  "Units": 1000, # $10
  "BalanceWeight": 1000 # 1000円
}
```

国際電話の料金を設定します。

10. 配置速率限制

配置速率限制

```
# 配置速率限制
"RateIncrement": "1s"

# 配置速率限制6s
"RateIncrement": "6s"

# 配置速率限制30s
"RateIncrement": "30s"

# 配置速率限制1024KB
"RateIncrement": "1024"
```

11. 配置过期时间

配置过期时间

```
# 配置过期时间
"ExpiryTime": "*month"

# 配置过期时间/30
"ExpiryTime": "+720h" # 30

# 配置过期时间/90
"ExpiryTime": "+2160h" # 90

# 配置过期时间/15
"ExpiryTime": "+360h" # 15
```

配置

配置

配置

関数

- 関数UUID
- 関数名
- 関数説明

関数 関数名関数説明関数説明

関数関数関数関数関数

関数 関数関数関数関数

関数

- 関数関数
- 関数関数ID

関数 関数関数関数ID関数関数関数

関数関数関数関数

関数 関数*topup関数関数関数

関数

- 関数関数ID関数関数ID
- 関数*topup_reset関数*topup

関数 関数ID関数関数関数

関数関数関数関数関数

関数 関数関数関数関数

関数

- 関数関数
- 関数関数関数関数関数

API 호출 시 요청 헤더에 +Xh 추가

API 호출 시 요청 본문

API 호출 시 요청 파라미터

API 호출 시 요청 데이터

1. DestinationIDs - API 호출 시 요청 본문
2. API 호출
3. API 호출 결과

API 호출

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

# API 호출 결과
# - API 호출
# - DestinationIDs
# - API 호출 결과
# - API > 0
```

API 호출 시 요청 본문에 1 추가

API 호출 시 요청 본문

API 호출 시 요청 파라미터

API 호출

1. API 호출 결과
2. API 호출
3. API 호출

API 호출

- 設定可能なパラメータ
- 設定可能なパラメータ
- 設定可能なパラメータ

設定可能なパラメータ

設定可能なパラメータ

設定可能なパラメータ

設定可能なパラメータ

1. PLMN設定 - PLMN
2. 設定可能なパラメータ
3. DestinationIDs設定 PLMN

設定可能なパラメータ

```
# PLMN CDR
# PLMN
"DestinationIDs": "Dest_PLMN_US_All" # mcc310.mnc004
```

設定可能なパラメータ

設定可能なパラメータ

設定可能なパラメータ

設定可能なパラメータ

1. DestinationIDs設定 *any
2. 設定可能なパラメータ
3. 設定可能なパラメータ

設定可能なパラメータ

- DestinationIDs: "*any"
- 設定可能なパラメータ

- 0000000000000000000010000

000 003000 000000000000

000000000000

000 0000000000000000

00000

1. 0000000000000000+100000000
2. 000000000000
3. 0000000000000000

00000

- 00CDR00000000000
- 000000000000000000
- 00NANP00000000000000000000

000 CGRateS Destinations 000000000000

00000

00000

- 0000 - 00000000CGRateS000000000000000000
- CGRateS Destinations - 000000000000000000PLMN000

000000

- 00Playbooks000000 - 0000000000000000000000
- Ansible Playbooks00 - Playbook000000000000000000
- 000000 - OmniCRM00000000

CGRateS

OmniCRM と CGRateS を連携して、
PLMNごとのSMS/通話

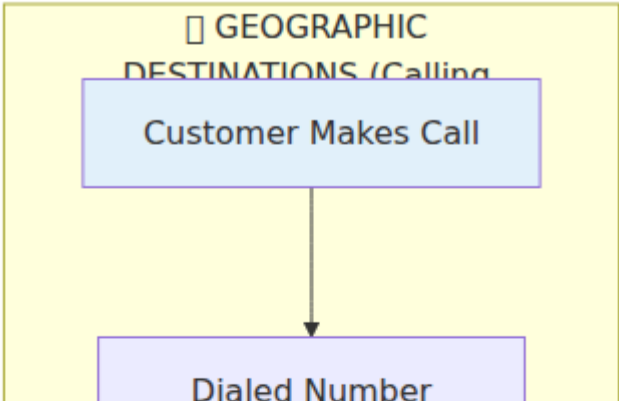
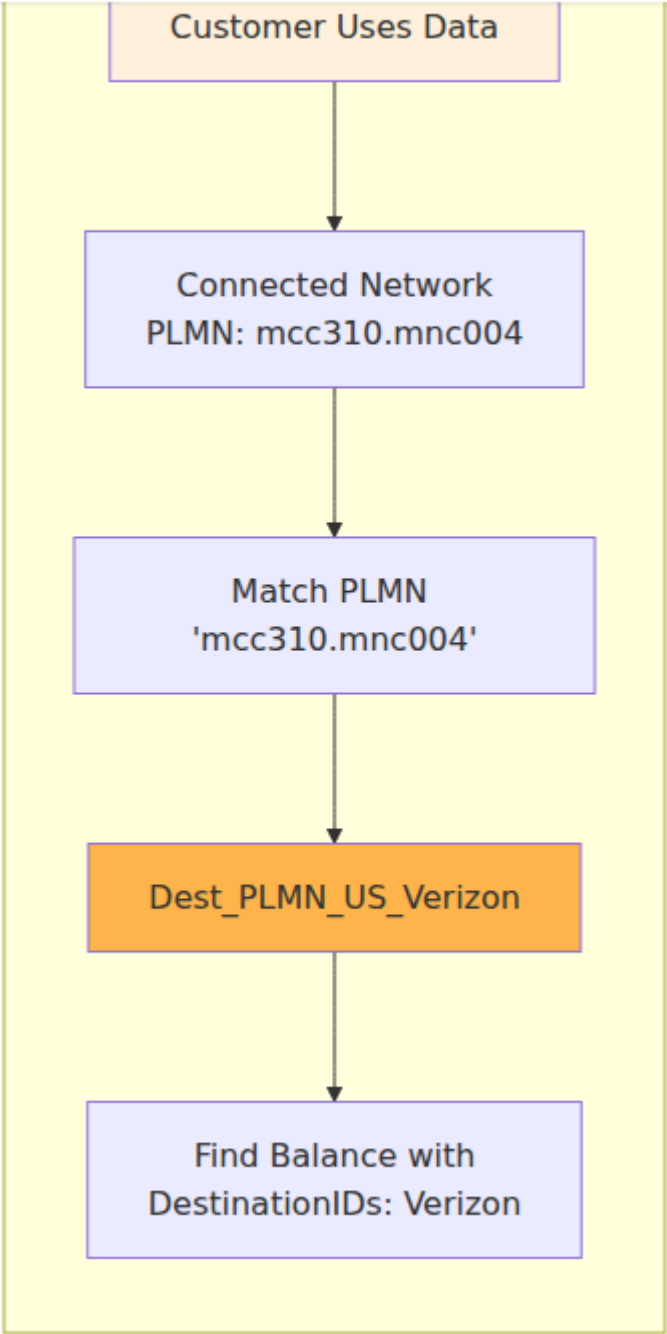


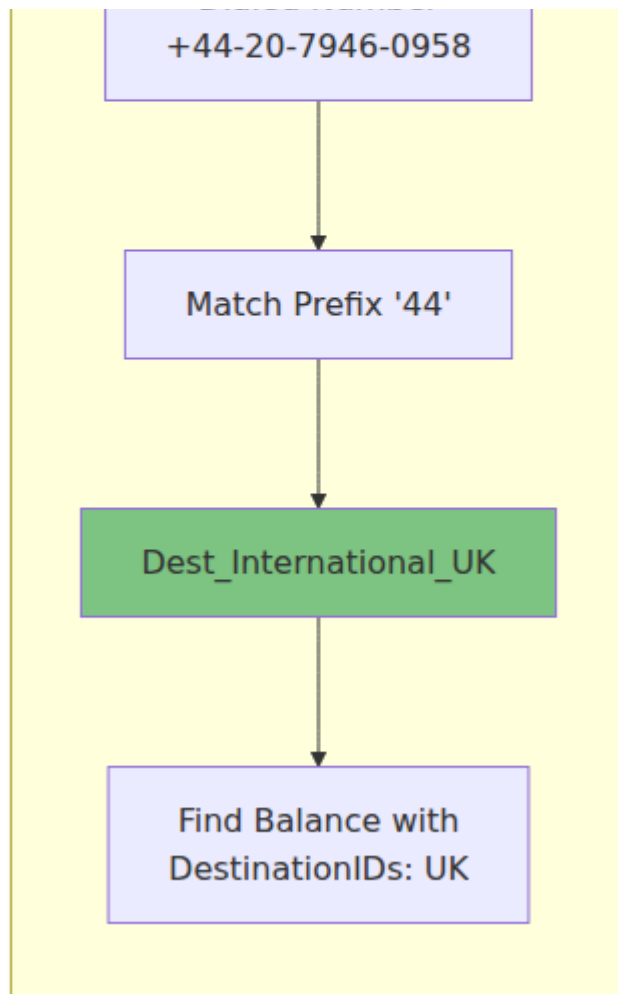
CGRateS は、SMS/通話の

- SMS - SMS/通話の
- **PLMN** - Verizon/Vodafone UK

CGRateS は、

- SMS/通話 →
- PLMN → PLMN





□□□□

- □□/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 - /

[illegible]

- **MCC** = 0000003 0000
- **MNC** = 0000002-3 0000???

PLMN

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

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

□□□□□□□□□□

□□□□□□□□□□ SIM □□□□□□□□□□

```
# □□□□□□□□□□
# □□□□□□ PLMN - □□□□□ SIM □□□□□□
# □□□□□□□□□□□□□□□□□□

OCS_obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_OnNet",
        "Prefixes": ["mcc505.mnc057"] # □□□□□□□□□□□□□□ MNC 505
    }]
})
# □ mcc505.mnc057 □□□□□□□□□□ PLMN □□
# □□□□□□□□□□□□□□□□□□□□
# □□□□□□□□□□
```

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"
        ]
    }]
})

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

# MCC310311
#   mcc310.mnc004Verizonmcc310.mnc410AT&Tmcc311.mnc580US
#   Cellular
#
```

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"]
    }]
})

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

“”“”

```

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

# 1
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_NorthAmerica",
        "Prefixes": [
            "mcc310", "mcc311", "mcc312", "mcc313", "mcc316", #
            "mcc302", "mcc334" #
        ]
    }]
})
# MCC
# " "

# 2
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_Europe",
        "Prefixes": [
            "mcc234", #
            "mcc208", #
            "mcc262", #
            "mcc222", #
            "mcc214", #
            "mcc228" #
        ]
    }]
})

# 3
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{

```

```

        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_AsiaPacific",
        "Prefixes": [
            "mcc505", # []
            "mcc460", # []
            "mcc454", # []
            "mcc440", # []
            "mcc520", # []
            "mcc525", # []
            "mcc530" # []
        ]
    }]
})

# []
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Roaming_All",
        "Prefixes": ["mcc"] # [] PLMN[]
    }]
})

```

[]

- [] Verizon 5GB → Dest_PLMN_US_Verizon
- [] 10GB → Dest_PLMN_US_All
- [] 15GB → Dest_PLMN_Zone_Europe
- [] 1GB → Dest_Roaming_All

項目	単位	制限	備考
通話	分/回線使用時間	“100分間使用可能”	SMS
PLMN	データ使用量	“100GB まで”	
PLMN	データ使用量	“Verizon まで 5GB”	/SMS
PLMN	データ使用量	“5GB”	/SMS
PLMN	データ使用量	“10GB まで”	/SMS

□ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □

CGRateS の ID は PLMN の 3 桁の後に続く
16桁の数字

□□□□□

- 0 00000000000000000000000000000000
- 0 00000000000000000000000000000000
- ⚠ 000000000000 - 00000000000000000000 LoadTariffPlanFromStorDb 000

1

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```

# 0000 - 0 3 000
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_PLMN_Zone_Europe",
        "Prefixes": [
            "mcc262", # 00
            "mcc208", # 00
            "mcc222"  # 000
        ]
    }]
})

```

00 **Dest_PLMN_Zone_Europe** 000000 000000000000000000

0000000000000000


```

# 設定PLMN
OCS_Obj.SendData({
  'method': 'ApierV2.SetTPDestination',
  'params': [{
    "TPid": tpid,
    "ID": "Dest_PLMN_Zone_Europe",
    "Prefixes": [
      "mcc208", # 日本
      "mcc222", # 韓国
      "mcc214", # 台湾
      "mcc234" # ドイツ
      # 中国mcc262
    ]
  }]
})

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

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

```

このコードを実行すると、OCSに「Dest_PLMN_Zone_Europe」のPLMN設定が送信されます。

3分間/SMS 制限

この制限はSMSの送信回数です。

```

# 国内向けSMS送信
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_Domestic_All",
        "Prefixes": [
            "1", # 国内向けSMS送信
            # 国内向けSMS送信 - "1" 国内向けSMS送信
        ]
    }]
})

# 海外向けSMS送信
OCS_Obj.SendData({
    'method': 'ApierV2.SetTPDestination',
    'params': [{
        "TPid": tpid,
        "ID": "Dest_North_America",
        "Prefixes": [
            "1" # 海外向けSMS送信
        ]
    }]
})

```

国内向けSMS送信 DestinationIDs: "Dest_Domestic_All" 国内向けSMS送信
"1" 国内向けSMS送信

国内向けSMS送信

国内向けSMS送信/SMS/国内向けSMS送信 国内向けSMS送信

国内 1 国内向けSMS送信

```
# 歐洲
{
  "BalanceId": "Europe_Data_5GB",
  "BalanceType": "*data",
  "DestinationIDs": "Dest_PLMN_Zone_Europe", # 歐洲區
  "Units": 5 * 1024 * 1024 * 1024
}

# 歐洲mcc214.mnc001
# 歐洲 Dest_PLMN_Zone_Europe
```

歐洲

- 歐洲區
- 歐洲
- 歐洲“歐洲”

歐洲 歐洲 Dest_PLMN_Zone_Europe 歐洲 DestinationIDs: "*any" 歐洲

歐洲 2歐洲

```
# 歐洲
{
  "BalanceId": "UK_Voice_100min",
  "BalanceType": "*voice",
  "DestinationIDs": "Dest_International_UK", # 歐洲 "44"
  "Units": 100 * 60 * 1000000000
}

# 歐洲 "*any" 歐洲

# 歐洲+61
```

歐洲

- 歐洲 "61"
- 歐洲 "44"
- 歐洲

```
00000000 DestinationIDs: "*any" 00000000000000000000000000000000
```

3

```
# 0000
# 00 1000000
{
  "BalanceId": "UK_Voice_100min",
  "BalanceType": "*voice",
  "DestinationIDs": "Dest_International_UK",
  "Units": 100 * 60 * 1000000000,
  "BalanceWeight": 1200
}

# 00 2000000000
{
  "BalanceId": "PAYG_Credit",
  "BalanceType": "*monetary",
  "DestinationIDs": "*any", # 0000
  "Units": 5000, # $50
  "BalanceWeight": 1000 # 00000
}

# 0000000000+610
```

111

- 0000
- 00000000 "61"
- PAYG 00000000 "*any"
- 0 PAYG 00000000

□ □ □ □ □

[illegible]

- 00000 PLMN 000000
- 000000000/000000
- 0000000000000000000000000000000000

- □□□□□□□□

[illegible]

--

1. 00000000 - 000000000000000000000000
2. 0000000000 - 00 LoadTariffPlanFromStorDb 0000000000000000
3. 00000000 = 0000 - 00000000000000000000000000000000
4. 0000 *any 000000 - 000000000000000000
5. 0000000000 - 000000 ??? 000000 ??? 00000000
6. 0000000000 - 000000000000000000000000

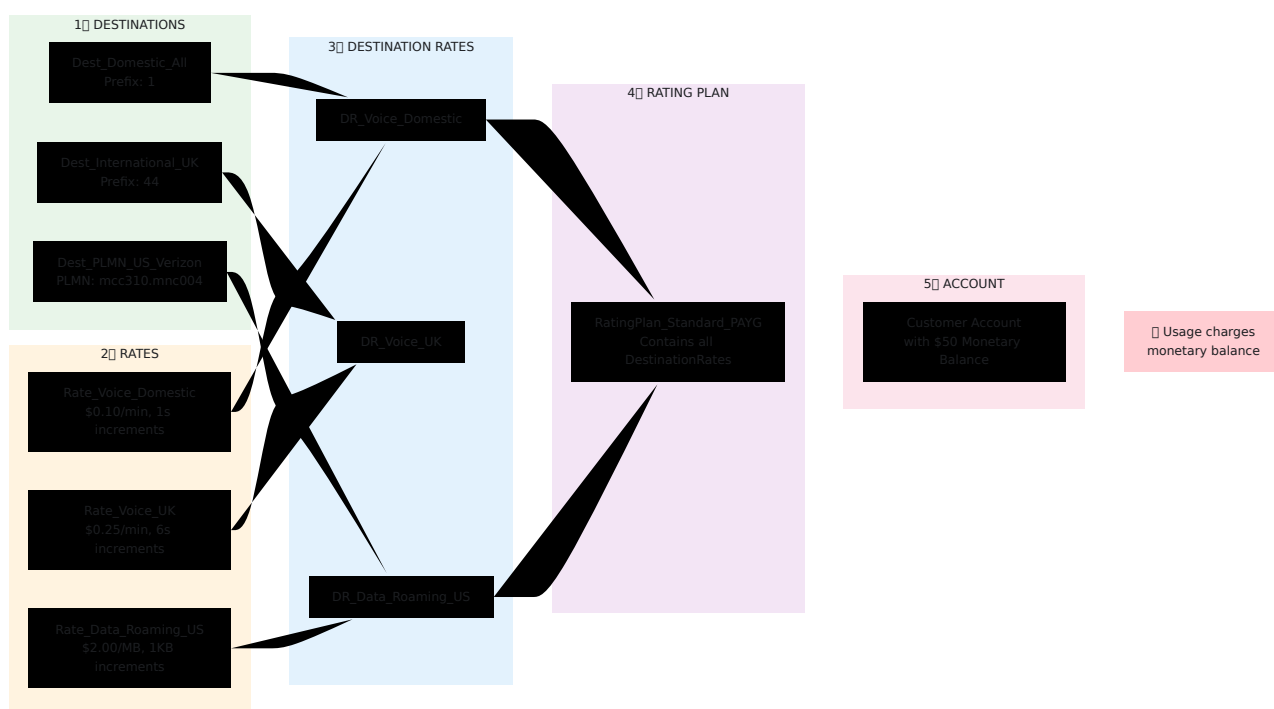
PAYG/

PAYG -

CGRateS CGRateS

- 國際行動電話網PLMN - 國際網
- 國際行動電話網國際網 - 國際網
- 國際行動電話網/國際網
- 國際網VIP國際網
- 國際網
- 國際網國際網國際網

PLMN CRM
CGRateS CRM



☐ SMS ☐ ☐ ☐ ☐

```

#
=====
# 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

```

# 国内目的地
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
        }]
    }]
})

# 英国目的地
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
        }]
    }]
})

# Verizon PLMN 国际漫游
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": [
            # □□□□□□□□□□ = □□□□
            {"DestinationRatesId": "DR_Voice_UK", "TimingId":
"*any", "Weight": 40},
            {"DestinationRatesId": "DR_Voice_Domestic",
"TimingId": "*any", "Weight": 20},

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

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

□□□□□□

□□□□□□□□□□ CGRateS □□

```
# 000000
OCS_obj.SendData({
    'method': 'ApierV1.LoadTariffPlanFromStorDb',
    'params': [{
        "TPid": tpid,
        "DryRun": False,
        "Validate": True
    }]
})

# 000000
OCS_obj.SendData({
    'method': 'CacheSv1.ReloadCache',
    'params': [{
        "Tenant": tenant,
        "DestinationIDs": ["*all"]
    }]
})
```

00000

- 0000 - 000000000000
- CGRateS 00000000 - 000000000000

0000

1. 0000000000

```
#  - 000000
"DestinationIDs": "Dest_PLMN_OnNet", # mcc505.mnc057

#  - 0000000000
"DestinationIDs": "Dest_PLMN_AU_All", # mcc505 AU 0000
```

2. 目的地別料金

```
# 1 - 目的地
Dest_PLMN_US_Verizon
Dest_International_UK
Dest_PLMN_Zone_Europe

# 2 - 料金
Dest_Data_1
Dest_Voice_Package
```

3. 国内料金

```
# 1 - 目的地
Dest_Domestic_All
Dest_Domestic_TollFree
Dest_Domestic_Premium

# 2 - 料金
Dest_Area_Code_212
Dest_Area_Code_213
Dest_Area_Code_214
```

4. 各 PLMN 別

各 PLMN 別料金表は MCC/MNC 別で提供

```

# Verizon Wireless
# https://www.mcc-mnc.com/ 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□□□□□□□□□□□□□□
# □□□□ 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

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

PLMN 設定

PLMN 設定の概要

設定

- PLMN 設定の PLMN ID は `Dest_PLMN_OnNet` である
 - PLMN ID +44-20-xxxx → PLMN ID "44"
- PLMN **DestinationIDs** 設定
 - PLMN ID "DestinationIDs": "Dest_International_UK" 及び "Dest_International_All"
- PLMN ID `ExpiryTime`
- PLMN ID `Units > 0`

PLMN 設定

- CGRateS** 設定 - PLMN 設定

- 0000 - 000000000000

Ansible Playbooks:

OmniCRM

Ansi

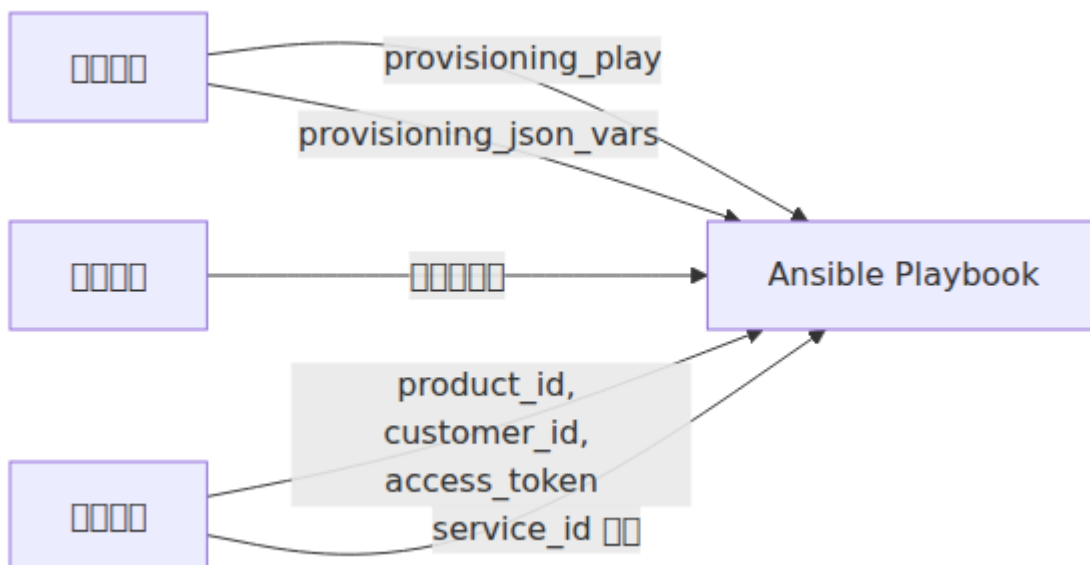
Playbooks

Playbooks OmniCRM playbook

Playbooks

OmniCRM

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



Playbooks ☐☐☐☐☐

playbook

概要

playbook の概要

- 概要
- IPTV の概要
- VoIP の概要
- 概要

概要

playbooks の概要

- CPE の playbook
- playbook
- playbook

概要

概要

- 概要
- 概要
- 概要

概要

playbooks

- 概要
- 概要
- 概要

概要

playbooks

- 概要
- 概要
- 概要

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

Play `OmniCRM-API/Provisioners/plays/` playbooks

111

- `play_simple_service.yaml` - 简单服务
- `play_topup_no_charge.yaml` - 充值不扣费
- `play_prov_cpe_mikrotik.yaml` - Mikrotik CPE provisioning

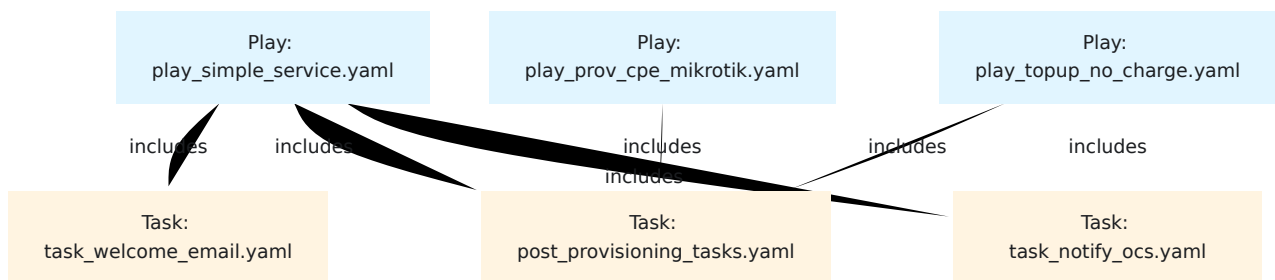
Task

`tasks` plays `Tasks` `task_`

111

- `task_welcome_email.yaml` - 歡迎郵件
- `task_activate_olt.yaml` - 激活 OLT
- `task_notify_ocs.yaml` - 通知 OCS

□□□□□□□□



```
# play_simple_service.yaml (Play)
- name: Install Play
  hosts: localhost
  tasks:
    - name: Install
      block:
        - name: Install
          uri: ...

        - name: Install
          uri: ...

    # Install
    - include_tasks: task_welcome_email.yaml

    # Install
    - include_tasks: post_provisioning_tasks.yaml
```

Playbook Install

OmniCRM playbooks are installed in the playbooks directory

Install

playbook install

```
- name: Playbook Install
  hosts: localhost          # OmniCRM on localhost
  gather_facts: no          # Install
  become: False             # Install

  tasks:
    - name: Install
      block:
        # Install

  rescue:
    # Install/Uninstall
```

#####

name

`playbook_description`

hosts: localhost

OmniCRM playbooks 在 localhost 上通过 API 而不是通过 SSH

gather_facts: no

Ansible 不收集事实

- 不收集事实
- 不收集变量
- 不收集主机信息

become: False

不通过 API 认证

#####

playbook 不认证

```
tasks:  
  - name: 创建 crm_config 文件  
    ansible.builtin.include_vars:  
      file: "../..crm_config.yaml"  
      name: crm_config
```

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

`crm_config.yaml` 文件中

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

#####

#####

#####

```
- name: 创建 OmniCRM 产品 product_id
  debug:
    msg: "创建 {{ product_id }}"
```

#####

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

API

```
- name: 调用 CRM API 获取产品
  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: 获取产品
  debug:
    msg: "获取产品 {{ api_response_product.json.product_name }}"
```


📖 **Playbook** 📖

📌📌📌📌📌

📌📌📌📌📌📌📌📌📌📌

```

- name: 0000 Playbook
  hosts: localhost
  gather_facts: no
  become: False

tasks:
  - name: 000
    block:

      # 1. 0000
      - name: 00 crm_config 000
        ansible.builtin.include_vars:
          file: "../../crm_config.yaml"
          name: crm_config

      # 2. 0000000
      - name: 0 CRM API 0000000
        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. 0000000
      - name: 0 CRM API 0000000
        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. 00000000000
      - name: 0000000
        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. 00000000

- name: 00 UUID
set_fact:
 uuid: "{{ 99999999 | random | to_uuid }}"
- name: 0000 UUID
set_fact:
 service_uuid: "Service_{{ uuid[0:8] }}"

6. 000000000000

- name: 0 OCS/CGRateS 000000
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: 00 OCS 000000
assert:

```

    that:
      - ocs_response.status == 200
      - ocs_response.json.result == "OK"

# 7. 余额充值
- name: 余额 0 充值
  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": "0000",
            "Value": 0,
            "ExpiryTime": "+4320h",
            "Weight": 1,
            "Blocker": true
          }
        }]
      }
    status_code: 200
  register: balance_response

# 8. CRM 余额充值
- name: 余额充值ISO 8601 格式
  command: date --utc +%Y-%m-%dT%H:%M:%S%z
  register: current_date_time

- name: 余额 API 充值
  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.
- 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,
        "retail_cost": "{{ setup_cost | float }}"
      }
    return_content: yes
    register: transaction_response

# 10.
- include_tasks: post_provisioning_tasks.yaml

```

```

rescue:

# 00/0000
- name: 000000000000
  debug:
    var: hostvars[inventory_hostname]

- name: 0 OCS 000000
  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: 000000000000 CRM 000000
  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: 0000000000000000
  assert:
    that:
      - action == "deprovision"

```

□□/□□□□

□□□□□□□□□□□□□□□□

```

- name: 0000 Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: 00 crm_config 000
      ansible.builtin.include_vars:
        file: "../..crm_config.yaml"
        name: crm_config

# 1. 000000
- name: 0 CRM API 000000
  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. 0000000000000000
- name: 0 CRM API 000000
  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. 00000000
- name: 000000
  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. 充值话费
- name: 充值话费
  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: 验证充值结果
  assert:
    that:
      - action_response.status == 200
      - action_response.json.result == "OK"

# 5. 重置账户
- name: 重置账户 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. 删除过期数据
- name: 删除过期数据
  command: "date --utc +%Y-%m-%dT%H:%M:%S%z -d '+30 days'"

```

```

    register: new_expiry_date

- name: 更新服务到期日期
  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. 发送短信
- name: 发送短信
  uri:
    url: "http://sms-gateway/api/send"
    method: POST
    body_format: json
    body:
      {
        "source": "10086",
        "destination": "{{ customer_phone }}"
        "message": "您的服务将于{{ new_expiry_date.stdout
      }},
    }
    status_code: 201
    ignore_errors: True

```

CPE 设备

设备信息存储在ONT表中

```

- name: CPE 部署 Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: 部署 crm_config 模板
      ansible.builtin.include_vars:
        file: "../crm_config.yaml"
        name: crm_config

    # 1. 部署 CPE 模板
    - name: 部署 hostvars 中的 CPE 模板 ID
      set_fact:
        cpe_inventory_id: "{{ hostvars[inventory_hostname]['WiFi Router CPE'] | int }}"
      when: "'WiFi Router CPE' in hostvars[inventory_hostname]"

    # 2. 部署 CPE 模板
    - name: 部署 CPE 模板
      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. 部署 CPE 模板
    - name: 部署 API 模板
      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. 部署 CPE 模板
    - name: 部署 CPE 模板
      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. 管道

```

- name: 管道 CPE 管道
  set_fact:
    cpe_hostname: "CPE_{{ cpe_inventory_id }}"
    cpe_username: "admin_{{ cpe_inventory_id }}"

- name: 管道密码
  set_fact:
    cpe_password: "{{ lookup('pipe', 'cat /dev/urandom | tr -
dc a-zA-Z0-9 | head -c 16') }}"

```

6. 无线网络

```

- name: 无线网络 SSID
  set_fact:
    wifi_ssid: "Network_{{ cpe_inventory_id }}"

- name: 无线网络词库
  set_fact:
    word_list:
      - apple
      - cloud
      - river
      - mountain
      - ocean

- name: 无线网络 PSK

```

```

set_fact:
    random_word: "{{ word_list | random }}"
    random_number: "{{ 99999 | random(start=10000) }}"

- name: [] WiFi PSK
  set_fact:
    wifi_psk: "{{ random_word }}{{ random_number }}"

# 7. []
- name: []
  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: []
  template:
    src: "templates/cpe_router_config.j2"
    dest: "{{ config_dest }}"

# 8. []
- name: []
  ansible.builtin.slurp:
    src: "{{ config_dest }}"
  register: config_content

# 9. []
- name: [] CPE []
  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": "[]{{ lookup('pipe', 'date +%Y-%m-%d') }}"
  }
  status_code: 200

# 10. []
- name: []
  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": ""
          },
          "To": [{
            "Email": "support@example.com",
            "Name": ""
          }],
          "Subject": "CPE [] - {{ 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

```

[]

[] CGRateS ActionPlans []

```

# 00 playbook 000000000000

# 1. 000 auto_renew 00
- name: 000 auto_renew 0000
  set_fact:
    auto_renew_bool: "{{ (auto_renew | string | lower) in ['true',
'1', 'yes'] }}"

# 2. 0000000000
- name: 0000000000
  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\\\":\\\"0000\\\"}}
"
            }
          ]
        }]
      }
    status_code: 200
    register: action_response
    when: auto_renew_bool

# 3. 0000 ActionPlan
- name: 00000000 ActionPlan

```

```

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: `ActionPlan`

```

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: 删除行动计划 ActionPlan
  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

```

删除计划

删除计划 playbook 删除计划 plays 删除计划

删除计划

task_welcome_email.yaml - 删除计划

```

# [] [] [] [] [] [] [] [] [] [] play [] []
# - api_response_customer [] [] [] [] [] [] []
# - package_name [] [] [] [] []
# - monthly_cost [] [] [] [] []
# - setup_cost [] [] [] [] []

- name: [] [] [] [] [] [] []
  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_fact:
    email_subject: "[] [] [] [] [] [] [] []"
    email_from_name: "[] [] [] []"

- name: [] [] [] [] [] [] [] [] [] []
  loop: "{{ api_response_customer.json.contacts }}"
  set_fact:
    recipients: "{{ recipients + [{'Email': item.contact_email,
'Name': item.contact_firstname ~ ' ' ~ item.contact_lastname}] }}"

- name: [] [] [] [] [] [] [] [] [] []
  set_fact:
    first_contact: "{{
api_response_customer.json.contacts[0].contact_firstname }}"

- name: [] [] [] [] [] [] []
  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": "👋 {{ first_contact }}👋👋👋👋👋👋👋👋",
    "HTMLPart": "👋 {{ first_contact }}👋<br/><h3>👋</h3>
<br/>👋 {{ package_name }} 👋👋👋👋👋<br/>👋👋👋👋$ {{ monthly_cost }}
<br/>👋👋👋👋$ {{ setup_cost }}<br/>👋👋👋👋👋👋👋 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 - []
```

```
#  playbook  #
#
- include_tasks: task_notify_ocs.yaml
```

task_notify_ocs.yaml □□□□□

```
- name: 00 OCS 0000
  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: □□ SIM □□□ ID
  set_fact:
    sim_inventory_id: "{{ hostvars[inventory_hostname]['SIM Card']
| int }}"
  when: "'SIM Card' in hostvars[inventory_hostname]"

- name: □□ SIM □□□□□
  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: □□ SIM □□□□
  set_fact:
    iccid: "{{ sim_response.json.iccid }}"
    imsi: "{{ sim_response.json.imsi }}"
    ki: "{{ sim_response.json.ki }}"
```

□□□□□□□□

```
- name: SIM
  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
```

00000000

000000/000

```
- name: 0000000000ISO 8601
  command: date --utc +%Y-%m-%dT%H:%M:%S%z
  register: current_date_time

- name: 00000000
  set_fact:
    today: "{{ lookup('pipe', 'date +%Y-%m-%d') }}"
```

00000000

```
- name: 00 30 00000000
  command: "date --utc +%Y-%m-%dT%H:%M:%S%z -d '+30 days'"
  register: expiry_date

- name: 0000 90 0000
  command: "date --utc +%Y-%m-%d -d '+{{ days }} days'"
  register: future_date
  vars:
    days: 90
```

UUID

UUID 生成

- name: UUID
set_fact:
 uuid: "{{ 99999999 | random | to_uuid }}"
- name: 服务UUID
set_fact:
 service_uuid: "SVC_{{ uuid[0:8] }}"

密码

- name: 密码
set_fact:
 password: "{{ lookup('pipe', 'cat /dev/urandom | tr -dc a-zA-Z0-9 | head -c 16') }}"

密码生成

- name: 密码生成
set_fact:
 words:
 - alpha
 - bravo
 - charlie
 - delta
 - echo
- name: 密码生成
set_fact:
 word: "{{ words | random }}"
 number: "{{ 99999 | random(start=10000) }}"
- name: 密码生成
set_fact:
 passphrase: "{{ word }}{{ number }}"

CGRateS/OCS

```
- name: 
  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: 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",
        "Categories": "*any",
        "Balance": {
          "ID": "000",
          "Value": 10737418240,
          "ExpiryTime": "+720h",
          "Weight": 10
        }
      }]
    }
  status_code: 200
```

充值接口

```
- name: 充值接口
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: □□□□□□□□  
  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: 获取租户属性
  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: 设置手机号
  set_fact:
    phone_number: "{{ attr_response.json.result.Attributes |
    json_query('[?Path==\'*req.PhoneNumber\'].Value[0].Rules\') | first
    }}"
  when: attr_response is defined

```

获取租户属性

设置手机号

```

- name: 设置租户属性
  set_fact:
    monthly_cost: "{{ custom_cost | default(50.00) }}"

- name: 设置租户属性
  debug:
    msg: "租户 UUID 为 {{ service_uuid }}"
  when: service_uuid is defined

```

设置租户属性

```

- name: []
  include_tasks: configure_cpe.yaml
  when: provision_cpe | default(false) | bool

- name: []
  assert:
    that:
      - action != "deprovision"
  when: action is defined

```

[]

```

- name: []
  uri:
    url: "{{ endpoint }}"
    method: POST
  when:
    - service_uuid is defined
    - customer_id is defined
    - action != "deprovision"
    - enable_feature | default(true) | bool

```

[]

[]

```

- name: 充值接口
  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 接口

```

- name: 获取客户信息
  uri:
    url: "{{ crm_config.crm.base_url }}/crm/site/customer_id/{{ customer_id }}"
    method: GET
    headers:
      Authorization: "Bearer {{ access_token }}"
  register: sites_response

- name: 调试输出
  debug:
    msg: "[{{ item.address_line_1 }}]"
  loop: "{{ sites_response.json }}"

```

配置

ignore_errors

```
- name: 短信 SMS 网关
  uri:
    url: "http://sms-gateway/send"
    method: POST
    body: {...}
  ignore_errors: true
```

断言

```
- name: 短信 API 断言
  assert:
    that:
      - response.status == 200
      - response.json.result == "OK"
```

数据源配置

```
- name: 短信数据源配置
  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: 短信数据源配置
  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: □□□□□□
debug:
var: hostvars[inventory_hostname]
- name: □□□□□□
debug:
msg: "□□ UUID: {{ service_uuid }}"
- name: □□ API □□
debug:
var: api_response_product.json

□□

□□□□□□ API □□□

```
- name: □□□□
  uri:
    url: "{{ billing_endpoint }}"
    method: POST
    body: {...}
  register: response

- name: □□□□□□
  assert:
    that:
      - response.status == 200
      - response.json.result == "OK"
  fail_msg: "□□□□□□{{ response.json }}"
```

□□□

□□□□□□□□□□□□□□

```
# □□□□□□□□□□
- name: □□□□□□□□
  uri:
    url: "{{ ocs_endpoint }}/get_account"
    method: POST
    body: {"Account": "{{ service_uuid }}" }
  register: account_check
  failed_when: false

# □□□□□□□□□□
- name: □□□□
  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: □□□□ API  
  uri:  
    url: "https://api.example.com/endpoint"  
    method: POST  
    headers:  
      Authorization: "Bearer {{ access_token }}"  
  validate_certs: yes
```

□□

□□□□□□□□


```
# 15日間のリース
# 15日間リース 1台
# 月額リース料 50% 割引
- name: 15日間リース
  command: "date -d 'last day of this month' +%d"
  register: days_in_month

- name: 15日間リース
  command: "date +%d"
  register: current_day

- name: 15日間リース
  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) }}"
```

Ansible Playbooks

Ansible

1. Ansibleのインストール
2. Ansibleのインストール
3. Ansibleのインストール API
4. Ansibleのインストール
5. Ansibleのインストール `action: "deprovision"` のインストール

Ansible playbook

```

- name: 部署应用
  hosts: localhost
  gather_facts: no

  tasks:
    - name: 部署应用
      assert:
        that:
          - product_id is defined
          - customer_id is defined
          - access_token is defined
        fail_msg: "部署失败"

    - name: 调用 API 接口
      uri:
        url: "http://localhost:5000/crm/health"
        method: GET
      register: health_check

    - name: 验证接口
      assert:
        that:
          - health_check.status == 200

```

部署应用

部署应用

```

# 部署应用 - 部署应用
customer_id: "{{ customer_id }}"

# 调用 API 接口 - 调用 API 接口
customer_id: {{ customer_id | int }}

```

部署应用

Ansible 簡介

Ansible playbooks 介紹

簡介

OmniCRM 系統部署

簡介

系統部署環境：SIM 卡 IP 地址

簡介

OmniCRM 系統部署

- product_id - 產品 ID
- customer_id - 客戶 ID
- service_id - 服務 ID / 類型
- access_token - API 訪問 JWT

Ansible Playbooks

Ansible Playbooks 介紹 rescue 操作

- 部署環境
- 系統部署
- 系統部署
- 系統部署

部署環境

系統部署環境

1. 系統部署

系統部署環境

2. 系統部署

playbook 操作 action: "deprovision" 系統部署

Ansible

OmniCRM playbooks 部署和配置

```
- name: 部署 Playbook
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: 部署
      block:
        # 部署 OCS
        - name: 部署 OCS 服务
          uri: ...

        - name: 部署 CRM
          uri: ...
          register: service_creation_response

        - name: 部署
          uri: ...

      rescue:
        # 部署失败处理
        - name: 部署失败处理
          debug:
            var: hostvars[inventory_hostname]

        # 部署 OCS 失败处理
        - name: 部署 OCS 失败处理
          uri: ...

        - name: 部署 CRM 失败处理
          uri: ...

        - name: 部署失败处理
          assert:
            that:
              - action == "deprovision"
```

□□□□□□

□□ `play_simple_service.yaml` □□□□□□

```

rescue:
# 1. 管道
- name: 管道名称/ID
  debug:
    var: hostvars[inventory_hostname]

- name: 管道日期
  set_fact:
    today: "{{ lookup('pipe', 'date +%Y-%m-%d') }}"

# 2. 获取 CRM API 服务 ID
- name: 获取 CRM API 服务 ID
  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: 设置 api_response_service
  debug:
    var: api_response_service
  when: api_response_service is defined

# 3. 设置 service_uuid
- name: 设置 service_uuid
  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. 设置 OCS 配置
- name: 设置 OCS 配置
  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: 删除 OCS 租户
  debug:
    var: response
  when: response is defined

# 5. 删除 CRM 服务
- name: 删除 CRM 服务
  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: 删除 CRM 服务
  debug:
    var: delete_service_response
  when: delete_service_response is defined

```



```
# 6. 確認デプロvision
- name: 確認デプロvision“true”/確認デプロvision
  assert:
    that:
      - action == "deprovision"
    fail_msg: "確認デプロvision"
    success_msg: "確認デプロvision"
```

確認デプロvision

ignore_errors: True

確認デプロvision ignore_errors: True 確認デプロvision確認デプロvision確認デプロvision

確認

確認 when 確認デプロvision

when: service_creation_response is defined

確認

確認デプロvision確認デプロvision

1. 確認デプロvision確認デプロvision
2. 確認デプロvision
3. 確認 OCS/確認デプロvision
4. 確認デプロvision

確認

確認デプロvision

- 確認デプロvision action == "deprovision" → 確認
- 確認デプロvision action → 確認

確認デプロvision確認デプロvision

OCS/CGRateS 確認

```

- name: 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

```

- name: 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

```

```

- name: SIM 更新
  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: 删除 ActionPlan
  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

```

测试

```
- name: 部署API
  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
```

部署API

部署API

1. 部署API
2. 部署API playbook `action: "deprovision"`
3. Playbook 部署API
4. 部署API
5. 部署API

部署 API 部署

```
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

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: 環境構築
  debug:
    var: hostvars[inventory_hostname]
```

4. 環境構築

環境構築の準備作業

- 環境構築
- OCS 環境
- 環境構築

5. 環境構築

環境構築の準備作業

6. 環境構築

OCS 環境 `ReloadScheduler: true` CGRateS 環境

環境構築

playbook 環境

- 環境構築
- 環境
- 環境
- 環境
- 環境

API 環境

- CRM 環境
- OCS 環境
- 環境
- 環境
- 環境

Ansible

Ansible の **block/rescue** を使ったエラーハンドリングの例

例

```
- name: テスト
  block:
    - name: テスト 1
      uri: ...
      register: response

    - name: テスト 1
      assert:
        that:
          - response.status == 200

    - name: テスト 2
      uri: ...

  rescue:
    # エラーハンドリング
    # 1. エラーメッセージ
    # 2. ログ出力
    # 3. Playbook を再開
    - name: エラーハンドリング
      uri: ...
```

関連する記事: [concepts_provisioning](#)

OmniCRM API

OmniCRM 提供 RESTful API - 与您的 UI 集成

OmniCRM 提供 RESTful API

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 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”值
Bearer“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
- 管理 VLAN 地址
- 管理 Ansible 端

管理 IP 地址

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 地址

```
import requests

crm_url = 'https://yourcrm.com'
session = requests.Session()

print("Provisioning data to server: " + str(crm_url))

headers = {
    "Content-Type": "application/json"
}

# 00000000
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 設定 / 設定項目



Ansible 設定項目

SIM 設定項目

設定項目

設定項目

設定項目

設定項目

OmniCRM 設定項目

設定項目

OmniCRM 設定項目

設定項目

□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

環境構築

環境構築の目的は、開発環境を整えることです。

OmniCRM のインストールと設定を行います。

必要な依存関係をインストールします。

環境構築の完了を確認し、必要に応じて IP アドレスを設定します。

環境構築

環境構築の目的は、開発環境を整えることです。

OmniCRM のインストールと設定を行います。

必要な依存関係をインストールします。

環境構築

環境構築の目的は、開発環境を整えることです。

環境構築の目的は、開発環境を整えることです。

- 環境構築の目的は、開発環境を整えることです。 CRM のインストールと設定を行います。
- 環境構築の目的は、開発環境を整えることです。 必要な依存関係をインストールします。
- 環境構築の目的は、開発環境を整えることです。 SIM のインストールと設定を行います。

['SIM Card', 'Phone Number'] の CRM のインストールと設定を行います。

- Ansible Playbook のインストールと設定を行います。 Play Ansible のインストールと設定を行います。 service_id ICCID MSISDN のインストールと設定を行います。

□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□

category □□□□□□□□□□□□□□□□

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

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

```
provisioning_play Ansible
playbook
```

000- 00 00000000000000 - 0000 0 00 000000000000 - 00000000 playbook 0000 -
00000000000000000000000000000000

□□□□□□□□□□□□□□□□□□□□/□□□□□□□□□□□□□□□□

□ □ □ □

service_type 0000000000000000

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- [illegible]

[illegible]

- 
- 
- 
- 

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

□ □ □ □

[illegible]

Font Awesome

Font Awesome 5

- Font Awesome - Font Awesome 5
- Font Awesome - Font Awesome 5
- Font - Font

Font Awesome

- Font Awesome - Font Awesome 5
- Font Awesome - Font Awesome 5
- Font/Font - Font
- Font - Font \$
- Font - Font

□□□□□□

FontAwesome

000* 000000000000"00"0"00"0"wifi"0 * 00000000 * 000000000000 * 00000000
0

□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□□

- 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 入門

□□□□□

- **Play**
 - Ansible playbook `yaml` `yaml`
 - `OmniCRM-API/Provisioners/plays/`
 - `yaml`
- **JSON**
 - `JSON` Ansible playbook `yaml`
 - `yaml`
 - Playbook `customer_id``product_id``service_id``access_token`
- `yaml`
 - `yaml`
 - `SIM` `IPv4`
 - `/`
 - `ID` `yaml`

□ □ □ □ □ □ □

[illegible]

□□□□□

- □□□□□
 - □□□□□□□□□□/□□
 - □□□□□□□□
 - □□□□□□□□□□
- □□□□□
 - □□□□□□□□□□□□/□□
 - □□□□□□□□□□
 - □□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□
 - □□ - □□□□□□□□□□□□
 - □□□□ - □□□□□□□□□□
 - □□□□□□□□

□□□□□

- □□□□□
 - □□ - □□□□
 - □□□□ - □□□□□□□“_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** - Unique identifier
- **product_name** - Product name
- **product_slug** - URL API friendly identifier
- **category** - Product category
 - **standalone** - Standalone product
 - **addon** - Addon product
 - **bundle** - Bundle product
 - **promo** - Promotion
- **service_type** - Service type (e.g., maintenance, repair, etc.)
- **comment** - Additional information
- **icon** - Font Awesome icon (e.g., fa-solid fa-sim-card)

Table

- **retail_cost** - Retail cost (0 or more)
- **wholesale_cost** - Wholesale cost
- **retail_setup_cost** - Retail setup cost
- **wholesale_setup_cost** - Wholesale setup cost
- **tax_percentage** - Tax percentage (10 or 10% or 12.5 or 12.5% or more)

□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□ 10%□□□□ \$50.00 → □□ \$5.00 □
- □□□□ 0%□□□□ → □□ \$0.00 □
- □□□□□□ → □□□□□□□□□□

□□□□□□□□□□

- **enabled** - □□□□□□□□□□□□□□□□ false □□□□□□□□
- **residential** - □□□□□□□□□□□□□□□□
- **business** - □□□□□□□□□□□□□□□□
- **customer_can_purchase** - □□□□□□□□□□□□□□ true□□□□□□□□□□ false□
- **available_from** - □□□□□□□□□□/□□□□□□
- **available_until** - □□□□□□□□□□□□/□□□□□□□□□□□□□□

□□□□□

- **contract_days** - □□□□□□□□□□□□□□□□30 □□□□□365 □□□□□0 □□□□□□□□□□
- **auto_renew** - □□□□□□□□
 - **prompt** - □□□□□□□□□□□□
 - **true** - □□□□□□□□□□
 - **false** - □□□□□□□□

- **allow_auto_renew** - 是否自动续费 false

其他属性

- **terms** - 服务条款
- **features_list** - 功能列表 Python 列表 ['Feature 1', 'Feature 2']

部署

- **provisioning_play** - 部署 Ansible playbook 文件 .yaml 文件路径 OmniCRM-API/Provisioners/plays/ 文件
- **provisioning_json_vars** - 部署 JSON 文件 Ansible playbook 部署参数 customer_id product_id service_id access_token
- **inventory_items_list** - 部署设备列表 ['SIM Card', 'Mobile Number'] 部署设备列表 ID 部署 playbook
- **relies_on_list** - 部署依赖 ID 部署依赖列表

其他属性

- **created** - 创建时间
- **last_modified** - 最后修改时间

□□□□□□

□□□□□□ **SIM**□

□ □ □ □ □ □ □ □ □ □ □

```
{
  "product_slug": "example-mobile-prepaid-mini",
  "product_name": "Example Mobile 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 Example Mobile 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"
}
```

□□□□□□□□□□

```
{
  "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 + □□□□□□□□□□□□□□□□□□□□□□□□□□□□

JSON Example

```
{
  "product_slug": "Mobile-Topup-5",
  "product_name": "PAYG £5 Topup",
  "category": "addon",
  "service_type": "mobile",
  "provisioning_play": "play_topup_monetary",
  "provisioning_json_vars": "{\"service_id\": \"\"}",
  "inventory_items_list": "[]",
  "retail_cost": 5,
  "retail_setup_cost": 0,
  "wholesale_cost": 0,
  "contract_days": 0,
  "residential": true,
  "business": false,
  "enabled": true,
  "customer_can_purchase": true,
  "icon": "fa-solid fa-coins",
  "features_list": "['£5 credit', 'Valid for 180 days']",
  "terms": "Valid for 180 days or until all credit is used. See our website for full rates",
  "comment": "£5 to use for Calls, SMS & Data"
}
```

Example of the JSON structure for the `service_id` parameter

Ansible

Ansible uses a YAML format to define playbooks. The API uses Ansible playbooks to define the service configuration.

Example

The following example shows the JSON structure for the `service_id` parameter.

1. **provisioning_json_vars** - The JSON structure for the `provisioning_json_vars` parameter.
2. **API** - The API endpoint for the `provisioning_json_vars` parameter.
3. **Example** - The example JSON structure for the `provisioning_json_vars` parameter.

4. ID를 인자로 inventory_items_list

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

[illegible]

□ □ □ □ □ □ □ □ □ □

```
"provisioning_json_vars": "{\"monthly_cost\": 50, \"data_gb\": 100}"
```

API

```
{
  "product_id": 10,
  "customer_id": 456,
  "monthly_cost": 45,
  "custom_param": "value"
}
```

Ansible `extra_vars`

```
{
  "monthly_cost": 45,      # 月間料
  "data_gb": 100,         # 月間 provisioning_json_vars
                             でのデータ容量
  "product_id": 10,       # 製品ID
  "customer_id": 456,     # 顧客ID
  "custom_param": "value", # カスタムパラメータ
  "access_token": "eyJ..." # アクセストークン
}
```

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

□ □ □ □ □ □ □ □

- `access_token` - API 호출을 위한 CRM 이 JWT 토큰을 IP/API 호출을 위한 토큰으로
변환하여 `refresh_token` 으로

- `initiating_user` - 启动用户 ID
- 产品 ID `product_id` 客户 ID `customer_id` 服务 ID `service_id`

示例

```
inventory_items_list: ["SIM Card", "Mobile Number"]
```

1. 启动/API 用户 - 启动用户 ID
2. 产品 ID - 产品 ID
3. **Playbook** 客户 ID - 客户 ID 通过 playbook 调用 CRM API

示例- SIM 客户 ID 789 - 产品 ID 101

通过 playbook 调用- SIM Card: 789 - Mobile Number: 101

通过 playbook 调用 ID 通过 CRM API 调用 ICCID IMSI MSISDN

Ansible 示例

示例 `extravars` 通过 Ansible playbook 调用 playbook

```
{{ variable_name }} 通过 playbook 调用 {{ product_id }}
{{ customer_id }} {{ monthly_cost }}
```

示例

示例

- `product_id` - 产品 ID
- `customer_id` - 客户 ID
- `service_id` - 服务 ID
- `access_token` - API 访问令牌
- `provisioning_json_vars` 变量
- API 调用

API 仕様

POST /api/v1/service_id: 123/playbook

- product_id: 45
- customer_id: 456
- service_id: 123
- access_token:
- provisioning_json_vars

playbook

- CRM API
- UUID
- OCS UUID
- CRM

API 仕様

- product_id -
- customer_id -
- ID SIM Card Mobile Number
- access_token - API

- product_id -
- customer_id -
- service_id - ID
- access_token - API

service_id - playbook

```

0000000000??0000000000  playbook 0000000000000000000000000000000000

```

- `product_id` - 整数
- `customer_id` - 整数
- `service_id` - 整数
- 整数 ID モデム ルーター
- `access_token` - API 整数

```

playbook play_seniors_package IPTV

```

11

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```

    OCS
    CGRateS
    ActionPlans
    ThresholdS

```

□ □ □ □ □ □ □ □ □ □ □

[illegible]

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

□ □

1. 房型 - 是否住宅
 - 是否住宅 `residential = true` 且 `business = false` 为住宅
 - 是否 `business = true` 为商业
2. 套餐 - 套餐类型
 - 套餐类型 `category = standalone` 为独立套餐
 - 套餐类型 `category = addon` 为附加套餐
3. 折扣 - 折扣类型

- `enabled = true` - อนุญาตให้ใช้งานได้
- ฟิลด์ `available_from` & `available_until` ว่าง - ใช้งานได้ตลอดเวลา
- `customer_can_purchase = true` อนุญาตให้ลูกค้าซื้อได้ - อนุญาตให้ลูกค้าซื้อได้



API ฟิลด์ API ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

- ฟิลด์/ฟิลด์ (`/crm/product/`) - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
- ฟิลด์ (`/crm/product/paginated`) - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

ฟิลด์ `include_disabled=true` อนุญาตให้เรียกใช้ API ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

- ฟิลด์ - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
- ฟิลด์ `???` - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
- ฟิลด์ - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
- ฟิลด์ - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

1. ฟิลด์ - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
 - ฟิลด์ `service_type = "mobile"` ฟิลด์ `service_type = "mobile"` ฟิลด์
 - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
2. ฟิลด์ - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
 - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API/ฟิลด์
 - ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API
3. ฟิลด์ที่จำเป็น - ฟิลด์ที่จำเป็น `residential` & `business` ฟิลด์ที่จำเป็นสำหรับการเรียกใช้ API

???

service_type = "mobile"

- business = true category != "addon"
- business = true category = "addon" service_type = "mobile"
-

- service_id** -
- customer_id** -
- product_id** -
- service_name** -
- service_type** - VoIP IPTV
- service_uuid** - OCS/CGRateS
- icon** - FontAwesome

- service_status** -
- service_provisioned_date** -
- service_active_date** -
- service_deactivate_date** -
- contract_end_date** -

- retail_cost** -
- wholesale_cost** -
- service_billed** - true

- **service_taxable** - 是否可課稅 true
- **invoiced** - 是否已開票
- **promo_code** - 優惠碼

其他

- **service_visible_to_customer** - 是否對客戶可見 true
- **service_usage_visible_to_customer** - 是否對客戶可見/是否可查 true

其他

- **provisioning_play** - 部署 Ansible playbook
- **provisioning_json_vars** - 部署 playbook 的變數
- **deprovisioning_play** - 取消部署 Ansible playbook
- **deprovisioning_json_vars** - 取消部署 playbook 的變數

其他

- **bundled_parent** - 是否為 bundle 的 parent service_id
- **site_id** - 地點/門市

其他

- **service_notes** - 服務備註
- **created** - 創建時間
- **last_modified** - 最後修改時間

其他

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

□ □ □ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □

- service_id[]customer_id[]product_id
- service_uuid[][][][][]
- service_provisioned_date
- provisioning_play[]provisioning_json_vars
- deprovisioning_play[]deprovisioning_json_vars
- bundled_parent
- invoiced[][][][][]
- created[]last_modified[][][]

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

□□□□□□ Web UI □ API □ Ansible playbooks □□□□□□□□□□□□□□□□□□□□□□□□

□ □ □ □ □ □

1. 目標 - API 仕様を調査し、Ansible playbooks
2. 調査 - 仕様を調査し、“仕様”
3. 調査 - 仕様を調査し
 - 仕様/仕様
 - 仕様 `relies_on_list`
 - 仕様 `available_from/available_until`
 - `enabled` `customer_can_purchase`
4. 調査 - 仕様を調査し
5. 調査 - 仕様 `inventory_items_list` 仕様 SIM 仕様
6. 調査 - 仕様“仕様”API 仕様

API with Ansible

Introduction

1. Overview

API Introduction

- `provisioning_play` - Ansible playbook `play_psim_only`
- `provisioning_json_vars` - Provisioning JSON vars
- `customer_id` - Customer ID
- `product_id` - Product ID
- `service_id` - Service ID
- `inventory_id` - Inventory ID

2. Steps

Steps to follow

1. Set `provisioning_json_vars` vars
2. Set vars
3. Set vars
 - `access_token` - API token CRM JWT
 - `initiating_user` - Initiating user ID
 - `customer_id` `product_id` `service_id`
4. Set `{inventory_type: inventory_id}` vars

Conclusion

```

{
  "customer_id": 123,
  "product_id": 456,
  "service_id": 789,          # 
  "SIM Card": 1001,          # 
  "Mobile Number": 1002,     # 
  "monthly_cost": 30,        # provisioning_json_vars
  "data_gb": 50,             # provisioning_json_vars
  "access_token": "eyJ...",  # API 
  "initiating_user": 5       # 
}

```

3

Provision

- `provision_id` -
- `provisioning_play` - playbook
- `provisioning_json_vars` - JSON
- `task_count` - playbook YAML
- `provisioning_status` - 1 = 0 = 2 = 1
- `product_id` `customer_id` `service_id` -

4 **playbook**

API

1. `OmniCRM-API/Provisioners/plays/{playbook_name}.yaml` playbook YAML
2. `ansible_runner.run()`
 - `playbook` - YAML
 - `extravars` - Ansible
 - `inventory` - `'localhost,'`
 - `event_handler` -
3. playbook

5 `ProvisioningEventHandler`

Ansible Provision_Event

- event_name - playbook
- event_number -
- provisioning_status -
 - 0 = -
 - 1 = -
 - 2 = -
 - 3 = - playbook ignore_errors: true
- provisioning_result_json -

6 Ansible playbook Provisioners/plays/*.yaml

Ansible playbook

- {{ access_token }} /crm/product/product_id/{{ product_id }} GET
- /crm/customer/customer_id/{{ customer_id }} GET
- GET /crm/inventory/inventory_id/{{ inventory_id }} ICCID MSISDN
- API
 - HSS
 - IMS IP
 - CGRateS/OCS
 - ENUM
 -
- /crm/transaction/ POST
- OCS/CGRateS POST retail_setup_cost
- OCS - OCS/CGRateS POST UUID

8. 00000000 - 0 OCS/CGRateS 000000 ActionPlans 000000000000

9. 00000000 - 0 /crm/service/ 00 PUT/POST 0000 CRM 0000000000

```
{
  "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. 0000 - 0 /crm/inventory/inventory_id/{{ inventory_id }} 00 PATCH 000
000000“000”0000

11. 0000000000 - 000000000000000000000000

00 **7**0000000000

0 playbook 0000

- 000 Provision.provisioning_status 000 **0**0000
- 00000 Provision.provisioning_status 000 **2**00000000
crm_config.provisioning.failure_list 00000000
- 000000000000 ignore_errors: true 000000 **3**0000000000000000000000

0000000000 CRM 0000000000000000000000000000

0000000000000000000000000000000000

00000 category: standalone00

- 00 customer_id 0 product_id
- 0000000000SIM 0000000000000000

- API PUT `/crm/service/` 新增服務
- 新增HSS/OCS服務
- SIM 服務

category: addon

- `customer_id` `product_id` `service_id` 新增服務
- 新增服務
- OCS 服務
- OCS 服務
- 服務
- 服務

category: bundle

- 服務
- 服務
- 服務 + 服務 + 服務
- 服務
- CRM 服務
- 服務 + IPTV + VoIP 服務

Playbook 新增

playbook 新增

1. `OmniCRM-API/Provisioners/plays/{playbook_name}.yaml`
2. **Ansible** `extravars` `{{ variable_name }}`
3. `Authorization: Bearer {{ access_token }}` **API**
4. `rescue` `ignore_errors`
5. 服務
6. 服務
7. `fail` 服務

Playbooks 簡介

每個 playbook 包含以下

- `customer_id` - 客戶 ID
- `product_id` - 產品 ID
- `service_id` - 服務 ID/服務名稱
- `access_token` - 用於 CRM API 的 JWT 令牌
- `initiating_user` - 發起用戶
- 庫存 ID `{{ inventory_type }}: inventory_id`
- 配置 `provisioning_json_vars` 變量
- 其他配置

Playbooks 包含以下

- 獲取產品 `GET /crm/product/product_id/{{ product_id }}`
- 獲取客戶 `GET /crm/customer/customer_id/{{ customer_id }}`
- 獲取庫存 `GET /crm/inventory/inventory_id/{{ SIM_Card }}`
- 創建交易 `POST /crm/transaction/`
- 更新服務 `PUT /crm/service/`
- 更新服務 `PATCH /crm/service/{{ service_id }}`
- 更新庫存 `PATCH /crm/inventory/inventory_id/{{ inventory_id }}`

如何創建 Playbook

創建 Playbook 步驟

1. Playbook 包含 `customer_id` `product_id` `service_id` `access_token`
2. 獲取服務 `GET /crm/service/{{ service_id }}` 獲取 `service_uuid`
3. 獲取產品 `GET /crm/product/product_id/{{ product_id }}` 獲取 `product_id`
4. 向 OCS 系統 `POST to CGRateS` 傳遞 `retail_cost`
5. 向 OCS 系統 `POST to CGRateS` 傳遞其他配置
6. 向 CRM 系統 `POST /crm/transaction/` 傳遞配置
7. 完成

創建 `Provision` 或 `Provision_Event` 事件

OCS 簡介

OCS 是電信業者用來管理客戶的系統，它與 CRM 系統緊密結合，提供客戶服務人員查詢客戶的資料，如 CGRateS 的剩餘話費、流量、短信等。

- 查詢客戶 - 根據客戶 DID 查詢客戶資料
- 查詢客戶 - 根據客戶號碼 MB 查詢 SMS 記錄
- 查詢客戶 - 根據客戶號碼查詢剩餘流量、短信、話費
- 查詢客戶 - 根據客戶號碼查詢剩餘話費 \$30 或 10GB 流量
- 查詢客戶 - 根據客戶號碼查詢客戶狀態

CRM 系統與 OCS 系統緊密結合，提供客戶服務人員查詢客戶的資料。

系統架構

系統架構圖顯示 OCS/CGRateS 系統與 CRM 系統的交互。

API 接口

系統提供以下 API 接口：

1. **API 接口 - 查詢客戶** `GET /crm/service/{service_id}` 查詢客戶資料
2. 查詢客戶 - API 接口返回客戶資料 `service_uuid`
3. **CGRateS API 接口 - 查詢 CGRateS 剩餘話費**
 - i. **Get_Balance(service_uuid)** - 查詢客戶剩餘話費 `BalanceMap`
 - 返回客戶剩餘話費、流量、短信等資料
 - 返回客戶 ID 和客戶狀態 ID
 - 返回客戶剩餘話費 `custom_Name_hr`、剩餘流量 `custom_Expiration`、剩餘短信 `custom_Description_String`
 - ii. **Get_ActionPlans(service_uuid)** - 查詢客戶剩餘話費
4. 查詢客戶 - CGRateS 剩餘話費

```

{
  "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. 서비스 사용량 - 서비스 사용량

- **ServiceUsage.js** - 서비스 사용량 3 서비스
- **UsageCard.js** - 서비스 사용량
- **UsageProgress.js** - 서비스 사용량/서비스 사용량
- 서비스 사용량 서비스 사용량

서비스 사용량

BalanceMap 서비스 사용량

CGRateS 項目

- **ID** - 識別子 "DATA_10GB_2025_01"
- **Value** - 値
 - 5368709120 = 5 GB
 - 3600 = 1 時間
 - 100 = 100 分
 - 25.50 = \$25.50
- **ExpirationDate** - ISO 8601 形式
- **Weight** - 重み
- **DestinationIDs** - 宛先 ID ["AU", "INTERNATIONAL"]

CRM 項目

- **custom_Name_hr** - ID
- **custom_Expiration** - "2025 年 1 月 15 日" "11 月"
- **custom_Description_String** - 説明
 - "5 GB" "10 GB"
 - "60 分" "時間"
 - "50 分"
 - "\$25.50"

項目

項目

- **service_visible_to_customer** - false
- **service_usage_visible_to_customer** - false

項目

- 項目/項目
- 項目/項目
- 項目/項目

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 說明

說明

說明 / 說明

說明 DID 說明 OCS 說明
Action_ServiceUUID_ServiceName_WhatitDoes

說明 \$60 的 GPON 說明 1TB 說明

Action_kj49-adsf-1234-9742_60_GPON_1TB_MonthlyExpiry

1. 說明 \$0
2. 的 CRM 的 HTTP POST 說明
3. 的 1TB 說明 1 的

說明 MRC 說明 ActionPlan_{ service_uuid
}_Monthly_Charge 的 ActionPlan 說明 ActionPlan 說明

說明 的 MRC

說明 ActionPlans 說明

說明

說明

說明/說明
說明

```

    ActionPlans  Ansible  ExecuteAction  
```

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

□□□□□□□□□□ \$10 □ PAYG □□□□□□□□□□□□□□□□□□□□

\$0

□ □ □ □ / □ □ □ □ □ □

ThresholdS

PAYG / 00000000 \$00

CRM OCS

CRM API Balances ActionPlans Actions ThresholdS

CRM API Actions Ansible Playbooks CRM Actions
Actions/Ansible Playbooks

0CS ActionPlans

ThresholdS

CRM ActionPlans

ActionPlans CRM

□□□□□□ ActionPlans

CRM ActionPlans

- ## 1. API - GET /crm/service/{service_id} API

- 查詢所有服務
- 取得 `service_uuid` OCS 服務 ID
- 取得 CGRateS API 查詢 UUID 的 ActionPlans
- 查詢取得 `Get_ActionPlans(service_uuid)` 的 CGRateS

2. **ActionPlan** 查詢 - 查詢所有 ActionPlan 查詢

```
{
  "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` - 查詢所有/ID/查詢所有查詢
- `PlanName` - 查詢所有查詢所有 playbook 查詢
- `NextExecTime` - 查詢 ActionPlan 的 ISO 查詢
- `custom_NextExecTime_hr` - 查詢所有查詢所有“11 天”“天”“2025 年 2 月 1 日”
- `ActionPlanId_split_dict` - 查詢 ActionPlanId 查詢所有查詢

3. 查詢所有 - ActionPlans 的 **ActionPlansTable** 查詢所有

查詢所有

- 查詢 - 查詢 ActionPlanId 的 `ProductID` 查詢
- 查詢 - 查詢所有查詢 `retail_cost`
- 查詢 - 查詢 `custom_NextExecTime_hr` 查詢所有查詢
- 查詢 - 查詢所有
 - 查詢 - 查詢所有查詢/查詢所有查詢
 - 查詢所有 - 查詢所有

刪除 ActionPlans

- 刪除“刪除”
- 刪除“刪除”

刪除 ActionPlans

刪除 ActionPlans

刪除 ActionPlan

1. ActionPlansTable “刪除”
2. “刪除”
3. DELETE /crm/oam/remove_action_plan/{action_plan_id}
4. API ocs.Remove_ActionPlan() CGRateS ActionPlan
5. “{service_id} ActionPlan {ActionPlanId}”
6. ActionPlan

刪除

1. ActionPlansTable “刪除”
2. “刪除”
3.
 - ActionPlanId product_id
 -
 - playbook
 -
- 4.
- 5.

刪除

(service_id, product_id) service_product_discount

-
- 0% 0

- 顧客サービス割引を適用する
- 顧客サービス割引を適用する 0% 割引を適用する

顧客サービス割引を適用する `customer_service.discount_*` 割引を適用する

割引を適用する

割引を適用する `auto_renew` 割引を適用する

- `auto_renew = "true"` 割引 - 割引を適用する ActionPlans
- `auto_renew = "prompt"` 割引 - 割引を適用する ActionPlans
- `auto_renew = "false"` 割引 - 割引を適用する ActionPlans

割引を適用する playbook 割引を適用する CGRateS 割引を適用する ActionPlan

- 割引を適用する ID 割引を適用する ActionPlanId
- 割引を適用する
- 割引を適用する
- 割引を適用する

ActionPlan 割引

ActionPlans 割引を適用する

割引

ServiceID_{service_uuid}__ProductID_{product_id}__CustomerID_{customer_id}

割引

ServiceID_abc-123-
def__ProductID_456__CustomerID_789__MonthlyRenewal

割引を適用する CRM

- 割引 ActionPlan 割引を適用する
- 割引を適用する

- 查詢所有
- 查詢所有

CRM 查詢所有 `ActionPlanId_split_dict` 查詢所有

查詢所有 **ActionPlans**

CRM 查詢所有 `ActionPlans` 查詢所有

查詢所有 `ServiceView.js` 查詢

- 查詢所有 `ActionPlans` 查詢所有
- 查詢所有
- 查詢所有

查詢所有

- 查詢所有
- 查詢所有
- 查詢所有

查詢

- 查詢 `ActionPlans` 查詢“查詢所有”
- 查詢所有

ActionPlans 查詢所有

CRM 查詢   查詢所有

`GET /crm/service/expiring?threshold=7`

查詢所有

- 查詢所有
- 查詢 `ActionPlans` 查詢所有
- 查詢所有 `ActionPlans` 查詢

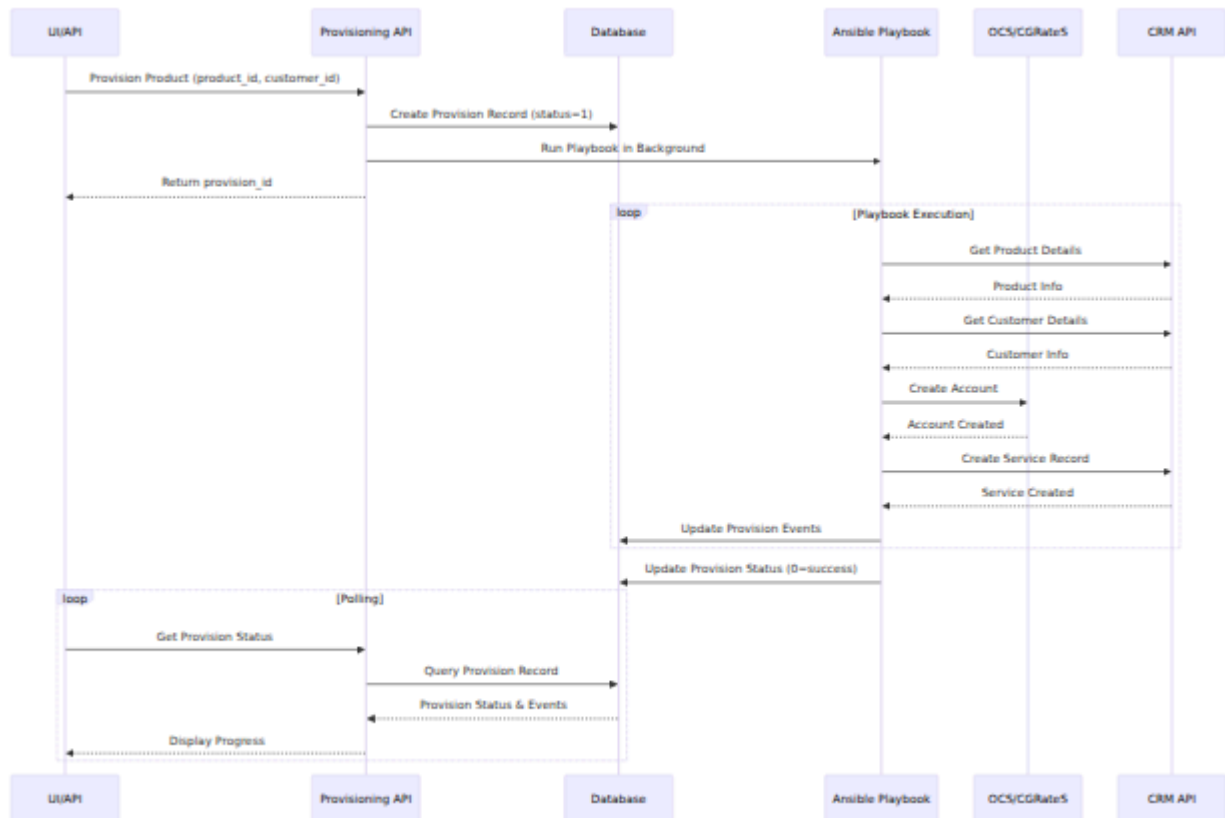
□□□□□□

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



OmniCRM 透过 **Ansible** 编排系统实现端到端的 eSIM 激活流程

透过 SIM 卡或 eSIM 激活流程





OmniCRM 的 Ansible 部署指南

简介

OmniCRM 的 Ansible 部署指南，旨在帮助用户快速部署 OmniCRM 的 OCS/CGRateS API 服务。

部署步骤如下：

1. 部署 - UI/API 服务
2. 部署 - OCS 服务

部署环境

部署环境要求如下：

- 至少 0 个 - 部署环境
- 至少 1 个 - 部署环境
- 至少 2 个 - 部署 - 部署环境
- 至少 3 个 - 部署 - 部署环境 `ignore_errors: true`

OmniCRM 的部署环境要求如下：

部署步骤

部署步骤如下：

部署环境

部署环境

- `provisioning_play` - Ansible 部署环境 .yaml 文件
- `provisioning_json_vars` - Ansible 部署环境 JSON 文件

- `inventory_items_list` - `['SIM Card', 'Mobile Number']`
- `product_id` `product_name` -

```
{
  "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
}
```

1. `provisioning_play`

`OmniCRM-API/Provisioners/plays/play_psim_only.yaml`

2. `extra_vars`

- i. `provisioning_json_vars` `{"iccid": "", "msisdn": ""}`
- ii. /API
- iii. `product_id` `customer_id`
- iv. `access_token` `refresh_token`

3. `inventory_items_list`

UI/API

- **SIM** - SIM

- `extra_vars` - `extra_vars` dictionary

Inventory ID `extra_vars` dictionary

```
extra_vars = {
    "product_id": 1,
    "customer_id": 456,
    "SIM Card": 789,          # SIM inventory_id
    "Mobile Number": 101,    # inventory_id
    "iccid": "",             # provisioning_json_vars
    "msisdn": "",            # provisioning_json_vars
    "access_token": "eyJ..."
}
```

4. `hostvars[inventory_hostname]` dictionary **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 `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. []

□□□□□□□□□□□□□□□□

- □□□□□ 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"
```

#####

- 当 ``action == "deprovision"`` 时返回 0
- 当 ``action`` 不等于 "deprovision" 时返回 2

#####

5. 测试

#####

```
- 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
```

#####

- 当 SIM 处于 (deprovision_subscriber: true) 状态时
- 当 SIM 处于 HSS 且 (deprovision_subscriber: false) 状态时

#####

#####

```
{
  "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

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 □□□□

55

- `Authorization: Bearer [API Key] IP [IP]`
- `CREATE_PROVISION`
- `initiating_user`
- `[IP]`
- `OmniCRM-API/Provisioners/plays/play_psim_only.yaml`

5. □□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □

```
# 字典
product_vars = json.loads(product['provisioning_json_vars'])

# 字典
request_vars = request_body_json

# 字典
system_vars = {
    'product_id': 42,
    'customer_id': 123,
    'access_token': auth_token, # 令牌
    'initiating_user': 7
}

# 字典
extra_vars = {**product_vars, **request_vars, **system_vars}
```

6.

□□□□□□□□□□ 1□□□□□□□

```
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. 準備

必要なものを準備する

- play_psim_only.yaml
- extra_vars
- ID 456 のプロビジョニング
- 成功を確認する

8. API 呼び出し

UI から provision_id を

```
{
  "provision_id": 456,
  "provisioning_status": 1,
  "message": "Provisioning job created"
}
```

9. UI 確認

UI から 3 分以内に GET /crm/provision/provision_id/456 を確認する

```

{
  "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 部署

部署环境

- 部署环境部署环境 Provision_Event 部署
- 部署环境部署环境0=部署2=部署3=部署部署部署部署 JSON
- UI 部署部署部署部署部署部署部署部署部署部署
- 部署部署部署部署部署部署部署部署部署部署

UI 部署

部署部署部署 1部署部署部署部署

- 部署部署部署 - 部署部署部署部署部署/部署/部署
- 部署 - 部署部署部署部署部署部署
- 部署部署 - 部署部署部署部署部署部署/部署部署部署部署

部署部署

部署部署部署部署12 部署部署 8 部署部署

✓ CRM API 8991101200003204510 ✓ 555-0123 OCS/CGRateS ...

11. UI

- provisioning_status: 0 -
- provisioning_status: 2 -

-
- X
- provisioning.failure_list

- initiating_user JWT ID
- API ID
- IP ID

- CREATE_PROVISION
- VIEW_OWN_PROVISION

Ansible CRM API

Ansible CRM API Bearer

access_token 〇〇〇〇〇〇〇〇〇〇〇〇 API 〇〇〇〇〇〇〇〇〇〇〇〇

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

5/5

Authorization Bearer
 API

2 API X-API-KEY

API Overview

1. `PUT /crm/provision/` `X-API-KEY: your-api-key...`
2. API `crm_config.yaml` API
3. **JWT**
- 4.
5. Ansible

□ □ □ □ □ □ □ □

API 0000000000 JWT00000 API 00000 JWT 000000000

- API 端
- 用户名 admin 和密码 JWT
- 用户名和密码 ID 和 JWT
- 用户名和密码 ID 和 JWT

5/5

API JWT

3IP

1. IP192.168.1.100
2. API IP `crm_config.yaml` `ip_whitelist`
3. JWT
- 4.
5. Ansible

IP JWT

API

```
- 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"
```

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.

測試驗證測試驗證 API 測試驗證測試驗證

測試驗證

- 測試驗證/測試驗證 Provision_Event 測試
- 測試驗證測試驗證測試驗證
- 測試驗證測試驗證
- UI 測試 GET /crm/provision/provision_id/<id> 測試

4. 測試

Ansible 測試測試測試

- 測試 API 測試測試
- 測試 API 測試測試
- 測試測試SIM 測試IP 測試測試測試
- 測試 OCS/OCS 測試測試
- 測試測試
- 測試 CRM API 測試測試
- 測試測試測試
- 測試測試測試測試/SMS

5. 測試

Ansible 測試 block 測試 rescue 測試測試

- 測試測試測試測試測試測試
- 測試 ignore_errors: true 測試測試 3測試測試測試
- 測試YAML 測試測試測試測試測試測試

測試測試測試

測試 play_simple_service.yaml 測試

```
- 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

```


- **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:

```

```

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_id | int }},
    "title": "{{ package_name }}",
    "retail_cost": "{{ monthly_cost | float }}"
  }

- name: Send Notification SMS
  uri:
    url: "http://sms-gateway/SMS/plaintext/{{ api_key }}"
    method: POST
    body_format: json
    body:
      {
        "source_msisdn": "YourCompany",
        "destination_msisdn": "{{ customer_phone }}",
        "message_body": "Thanks for topping up {{ monthly_cost
      }}!"
      }
    status_code: 201
    ignore_errors: True

```

□□□□

1. □ API □□□□□□□□□□
2. □□□□□□□□□□
3. □□ Stripe API □□□□□
4. □□□□□□□ OCS
5. □ CRM □□□□□
6. □□□□ SMS□□□ □ignore_errors: True□□□□□□□□□□□□□□

目標

OmniCRM から 1 台の IP に対して 1 台の service_id を取得する

前提条件

1. 1 台の IP に対して 1 台の service_id を取得する
2. IPTV サービスの 1 台の service_id を取得する
3. 1 台の IP に対して 1 台の service_id を取得する

実行方法

- 1 台の IP に対して 1 台の service_id を取得する
- 1 台の IP に対して 1 台の extra_vars を取得する
- 1 台の IP に対して 1 台の Provision を取得する

実行方法

実行方法

実行方法

実行方法 2

実行方法

- API を呼び出す
- assert: that: response.status == 200
- 1 台の IP に対して 1 台の service_id を取得する
- 1 台の IP に対して 1 台の extra_vars を取得する
- 1 台の IP に対して 1 台の Provision を取得する
- OCS/OCS を取得する

実行方法 3

実行方法

- 通过 SMS/email 通知
- 配置忽略错误 `ignore_errors: True`
- 通过  通知

配置

配置

- 配置 YAML 文件
- 配置
- 配置
- 通过 Ansible 配置

配置

- Ansible 配置
- 通过 `stdout` 配置
- 通过 `stderr` 配置
- 配置
- 配置

配置

配置 2 配置 `provisioning.failure_list` 配置 `crm_config.yaml` 配置

配置

- 配置
- 配置/配置
- 配置
 - 配置
 - 配置
 - 配置
- 配置/配置
- 配置 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

- 设置 **block/rescue** 环境变量
- 设置 **ignore_errors** 环境变量
- 设置环境变量 `debug` 用于调试
- 设置环境变量 `assert` 用于 API 断言
- 设置环境变量 `refresh_token`

环境变量

- 设置环境变量 `refresh_token`
- 设置环境变量 `IP` 用于 API 调用
- 设置环境变量 `refresh_token` 用于 API 调用

□□□□

- □□□□□□□□□□ customer_id□service_id □□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□ Provision_Event □□□□□□□□□□/□□□□

□□

- □□□□□□□□□□ customer_id□product_id□service_id
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□
- **IP** □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□

- □□□□□□□□□□ API □□□□□□□□
- □□□□□□UI □□ 2-5 □□□□□□□□□□
- □□□□□□□□ Ansible □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□

- concepts_ansible - □□ Ansible □□□□
- concepts_sim_provisioning - SIM □□□□□□□□ eSIM□
- concepts_api - CRM API □□□□□□□□
- concepts_products_and_services - □□□□□□□□
- administration_inventory - □□□□□□□□
- □□□□□□ - □□□□□□□□□□

SIM

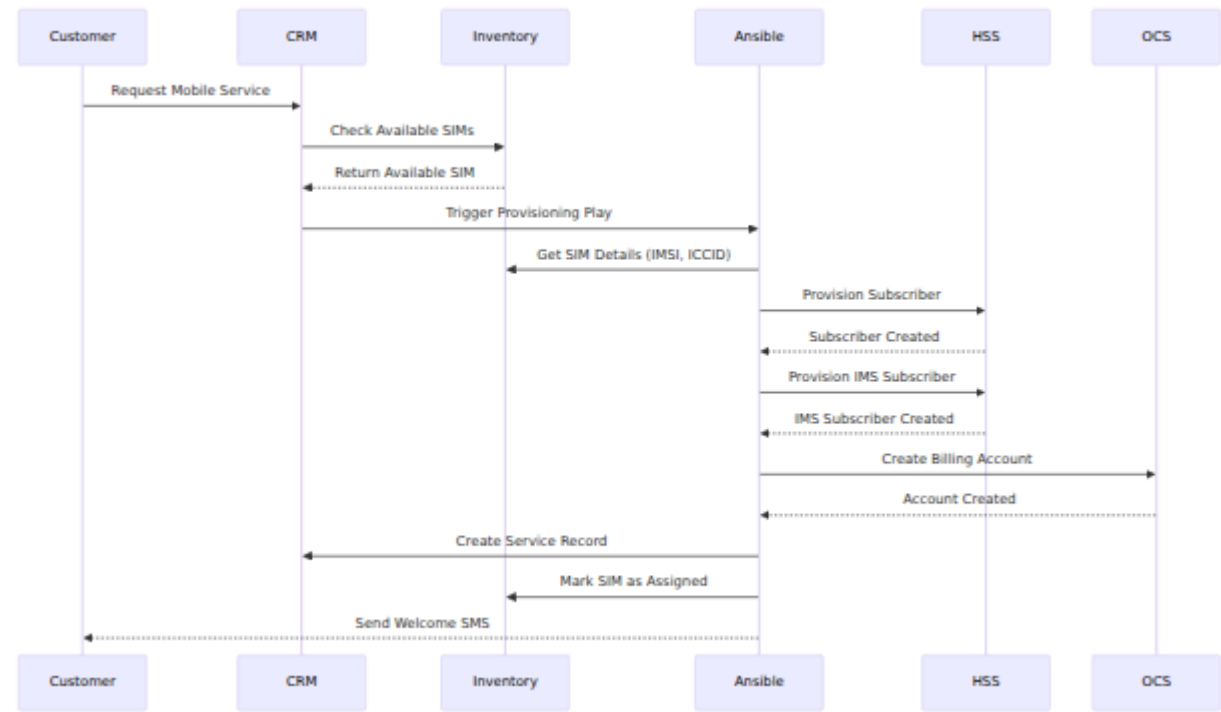
OmniCRM is a platform that allows MVNOs to manage SIMs and eSIMs. It provides a comprehensive set of tools for managing the entire SIM lifecycle, from provisioning to activation and deactivation.

OmniCRM is built using Ansible, a powerful automation tool that simplifies the management of infrastructure and services.

Flow

OmniCRM SIM Provisioning Flow

- Customer - Request Mobile Service
- HSS/IMS - Provision Subscriber
- OCS - Create Billing Account
- Ansible - Trigger Provisioning Play
- Inventory - Mark SIM as Assigned



📱SIM📱📱

📱SIM📱📱📱📱📱📱SIM📱OmniCRM📱📱📱📱📱📱📱HSS/IMS📱📱📱📱📱

📱SIM📱📱

1. 📱📱

SIM📱📱📱📱📱📱📱

- **ICCID**📱📱📱📱📱📱 - 📱SIM📱📱
- **IMSI**📱📱📱📱📱📱 - 📱📱📱📱

📱📱📱📱📱📱📱

- `itemtext1`: ICCID
- `itemtext2`: IMSI
- `itemtext3`: SIM📱📱📱/eSIM📱 - 📱

📱📱SIM📱📱📱

```
{
  "inventory_id": 1001,
  "item": "SIM Card",
  "itemtext1": "8961234567890123456", // ICCID
  "itemtext2": "310120123456789",    // IMSI
  "itemtext3": "Physical",            // SIM Type
  "item_location": "Warehouse A, Shelf 3",
  "item_state": "New",
  "wholesale_cost": 2.50,
  "retail_cost": 10.00
}
```

📱📱📱📱📱itemtext1-20📱item_state📱📱📱📱management_url📱📱📱📱📱📱📱
- 📱📱📱📱📱

📱📱📱📱📱

SIM📱📱📱📱📱**HSS AuC📱📱📱📱**📱📱📱📱CRM📱📱📱

- **Ki** - HSS
- **OPC** - HSS
- **PIN1/PIN2** - PIN HSS
- **PUK1/PUK2** - PIN HSS

CRM SIM ICCID/IMSI HSS SIM Ki/OPC

2.

1. SIM
2. MSISDN
3. `play_psim_only.yaml`

3. HSS

Ansible HSS

`auc_id` HSS

```
- name: Get AuC ID for IMSI
  uri:
    url: "{{ item }}/auc/imsi/{{ imsi }}"
    method: GET
  loop: "{{ hss_peers }}"
  register: auc_lookup

- name: Extract auc_id
  set_fact:
    auc_id: "{{ auc_lookup.results[0].json.auc_id }}"
```

`auc_id`

```
- name: Provision subscriber on HSS
  uri:
    url: "{{ item }}/subscriber/"
    method: PUT
    body_format: json
    body:
      enabled: true
      roaming_enabled: true
      auc_id: "{{ auc_id }}"
      msisdn: "{{ phone_number }}"
      imsi: "{{ imsi }}"
      ue_ambr_dl: 9999999
      ue_ambr_ul: 9999999
      apn_list: "1,2,3"
      default_apn: 1
  loop: "{{ hss_peers }}"
```

□□□□□□Ki□OPC□□□□HSS AuC□□□□□□□□CRM□□□□□□

4. IMS

IMS

```
- name: Create IMS subscriber for voice services
  uri:
    url: "{{ item }}/ims_subscriber/"
    method: PUT
    body_format: json
    body:
      imsi: "{{ imsi }}"
      msisdn: "{{ phone_number }}"
      msisdn_list: "{{ phone_number }}"
      ifc_path: "default_ifc.xml"
      sh_profile: "{{ sh_profile_xml }}"
  loop: "{{ hss_peers }}"
```

5.

OCS□□□□□□□□□□

- 電話番号
- IMSI/MSISDN電話番号
- 電話番号
- 電話番号
- 電話番号

6. 電話番号

番号

- CRM番号
- SIM番号customer_id
- SIM番号service_id
- 番号“番号”

SIM番号

SIM番号HSSAuC番号

- KiOPC4G/5G番号MILENAGE
- SIM番号HSS AuC
- SIM番号Ki/OPC
- 番号auc_id
- SIM番号HSS-番号
- CRM番号Ki/OPC

eSIM

eSIM番号SIM番号SIM番号SIM番号OmniCRM番号LPA番号
番号eSIM番号

eSIM

LPA番号

eSIM番号LPA番号

```
LPA:1$smdp.example.com$ACTIVATION-CODE-ABC123XYZ
```

111

- LPA:1 - LPA□□□□□
- smdp.example.com - SM-DP+□□□□□□□□□□□□□□
- ACTIVATION-CODE-ABC123XYZ - eSIM□□□□□□□□□□

555

OmniCRM LPA

- `management_url`
- UI
- eSIM
-

□□eSIM□□□□

```
{
  "inventory_id": 1002,
  "item": "eSIM",
  "itemtext1": "8961234567890123457",
  "itemtext2": "310120123456790",
  "itemtext3": "eSIM",
  "management_url": "LPA:1$smdp.example.com$ACTIVATION-CODE-ABC123XYZ",
  "item_location": "Virtual Inventory",
  "item_state": "New",
  "wholesale_cost": 0.00,
  "retail_cost": 0.00
}
```

UI

eSIM□□□□□

eSIM SIM

1. 查詢eSIM

查詢所有SIM卡或查詢SIM卡庫存數量或查詢eSIM

```
# 查詢eSIM
search_filters = {
    "customer_id": [None],
    "item_state": ["New"],
    "itemtext3": ["eSIM"] # 查詢eSIM
}
available_esim = search_inventory("SIM Card",
filters=search_filters)
```

2. HSS/IMS

eSIM卡與SIM卡

- 查詢IMS
- 查詢IMS
- 查詢Ki

3. 查詢

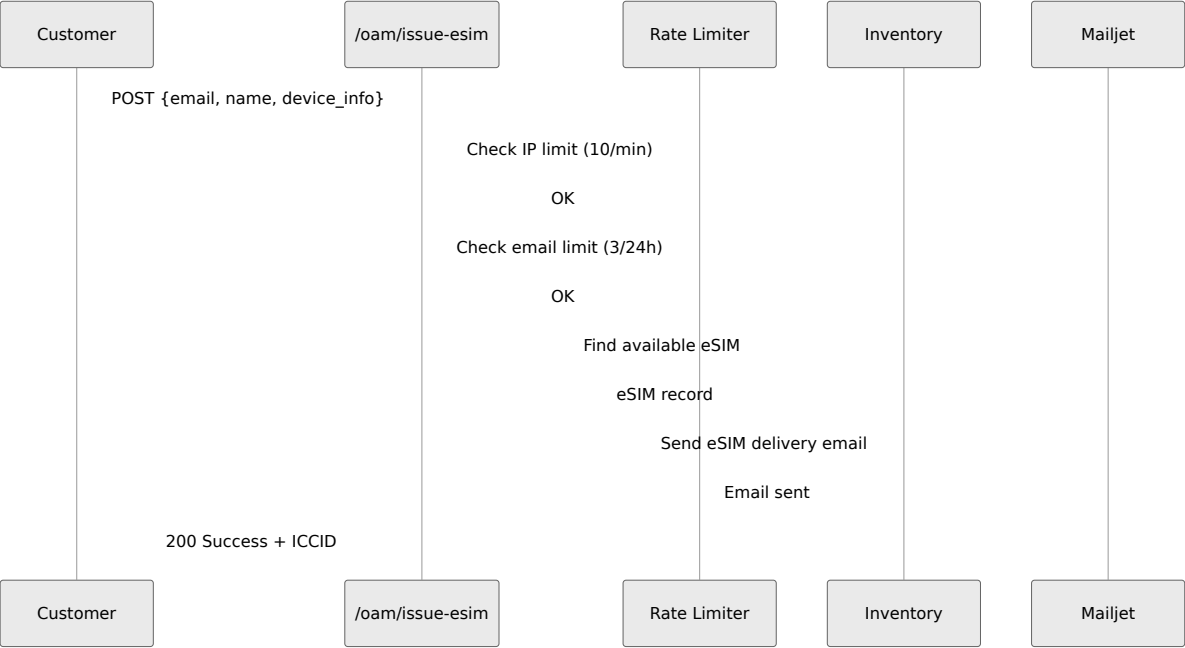
查詢

1. 查詢eSIM
2. 查詢
3. 查詢
4. 查詢SM-DP+查詢eSIM
5. eSIM

查詢eSIM API

OmniCRM API 查詢eSIM

POST /crm/oam/issue-esim



□□□□

項目	項目名	項目型	項目値	項目説明
email	項目名	項目型	-	項目説明 eSIM項目説明 項目説明
name	項目名	項目型	項目値	項目説明 項目説明 項目説明 項目説明
customer_id	項目名	項目型	null	項目ID項目 項目説明 項目説明
device_info	項目名	項目型	{}	項目説明 項目説明
device_info.compatibility_status	項目名	項目型	"unknown"	eSIM項目説明 supported項目 may-support unknown項目 項目
device_info.user_agent	項目名	項目型	項目値	項目説明 項目説明 項目

項目

項目説明

項目	単位	項目	説明
10回	1回	IPアドレス	同時に実行可能な回数
3回	24時間	同時実行数	同時に実行可能なeSIM数

エラーメッセージ: HTTP 429 `error_type: "RateLimitExceeded"`

原因

解決策: HTTP 200

```
{
  "result": "Success",
  "message": "eSIM request received. You will receive your eSIM QR code via email shortly.",
  "esim_id": 8472,
  "iccid": "8944200000001234567",
  "email_sent": true
}
```

項目	単位	説明
<code>result</code>	結果	HTTP 200の場合は <code>"Success"</code>
<code>message</code>	メッセージ	メッセージ
<code>esim_id</code>	ID	eSIM ID
<code>iccid</code>	ICCID	eSIM ICCID
<code>email_sent</code>	メール送信	メール送信の有無

備考

HTTP		
400	-	
429	RateLimitExceeded	
503	NoInventory	
500	-	

```
{
  "result": "Failed",
  "reason": "Error description",
  "error_type": "ErrorType"
}
```

Mailjet eSIM

- ICCID
- itemtext2
- /LPA itemtext3 management_url
-

crm_config.yaml Mailjet

```
mailjet:
  api_crmCommunicationEsimDelivery:
    from_email: "support@aimobile.ac"
    from_name: "Example Mobile"
    template_id: 1234567
    subject: "Your eSIM is Ready"
```

項目	型	必須	説明
from_email	文字列	○	送信元メールアドレス
from_name	文字列	○	送信元名前
template_id	整数	○	MailjetのIDを指定してHTMLを送信
subject	文字列	○	件名

送信元メールアドレスを指定してHTMLを送信

- 送信元名前
- eSIMの申し込みを促すメール
- 送信元アドレス
- iOS/Androidのアプリをダウンロード

送信元

eSIMの申し込みを促すメール

- 送信元 `esim_request`
- IDを指定して送信
- 送信元
- 送信元

送信元eSIMの申し込みを促す

4. HSSのeSIMの申し込み

HSS AuCのeSIMの申し込み `esim: true`

```
{
  "esim": true,
  "lpa": "LPA:1$smdp.example.com$ACTIVATION-CODE-ABC123XYZ",
  "iccid": "8961234567890123457",
  "imsi": "310120123456790",
  "ki": "00112233445566778899AABBCCDDEEFF",
  "opc": "FFEEDDCCBBAA99887766554433221100",
  "pin1": "1234",
  "pin2": "5678",
  "puk1": "12345678",
  "puk2": "87654321",
  "batch_name": "eSIM_Batch_2024_Q1",
  "sim_vendor": "Thales"
}
```

eSIM

eSIM

/OmniCRM-

API/Provisioners/customer_product_creation/import_keys_esim.py

1. Excel
2. CSV
3. HSS
4. CRM
5. eSIM

eSIM

SIM

SIM

SIM

```
{
  "item": "SIM Card",
  "itemtext1_label": "ICCID",
  "itemtext2_label": "IMSI",
  "itemtext3_label": "SIM Type",
  "wholesale_cost": 2.50,
  "retail_cost": 10.00
}
```

Ki OPC PIN PUK HSS AuC CRM

itemtext1-20 -

```
{
  "inventory_id": 4001,
  "item": "Mobile Number",
  "itemtext1": "+61412345678",
  "itemtext2": "Melbourne",
  "itemtext3": "Mobile",
  "item_location": "Australia - VIC",
  "item_state": "New",
  "wholesale_cost": 1.00,
  "retail_cost": 0.00
}
```

SIM

SIM

- SIM

- `product_id` - 產品ID
- `product_slug` - 產品代號
- `product_name` - 產品名稱
- `provisioning_play` - 產品配置
- `inventory_items_list` - 產品清單
- `inventory_items_list` - 產品清單
- `inventory_items_list` - 產品清單

Inventory Items List

Inventory Items List

```
{
  "product_id": 1,
  "product_slug": "Mobile-SIM",
  "product_name": "Mobile SIM Only",
  "provisioning_play": "play_psim_only",
  "inventory_items_list": ["SIM Card", "Mobile Number"]
}
```

Inventory Items List

1. UI
2. SIM
3. eSIM
4. ID
5. Ansible
6. API

Inventory Items List - ? ?

Inventory Items List

Ansible

CRM API Bearer API - ? ?


```

- 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 }}"
  register: sim_inventory_response

- name: Extract SIM identifiers
  set_fact:
    iccid: "{{ sim_inventory_response.json.itemtext1 }}"
    imsi: "{{ sim_inventory_response.json.itemtext2 }}"

- name: Get AuC ID from HSS (reference to authentication
credentials)
  uri:
    url: "{{ item }}/auc/imsi/{{ imsi }}"
    method: GET
  loop: "{{ hss_peers }}"
  register: auc_response

- name: Extract AuC ID
  set_fact:
    auc_id: "{{ auc_response.results[0].json.auc_id }}"

# ... provision to HSS/IMS/OCS ...

- name: Assign SIM 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 }}"
      service_id: "{{ service_id }}"
      item_state: "Assigned"

```

- SIM卡绑定
- 绑定ID卡
- 绑定银行卡
- 绑定其他SIM卡

SIM卡

OmniCRM系统支持SIM卡、eSIM卡

SIM卡

1. 绑定SIM卡

绑定SIM卡

- 绑定SIM卡
- 绑定SIM卡
- 绑定SIM卡

2. SIM卡

SIM卡

GET /auth/self-signup/validate-itemtext1/{sim_number}

绑定SIM卡

- SIM卡
- SIM卡
- SIM卡“”“”“”“”

3. SIM卡SIM卡

SIM卡SIM卡

```

def _populate_base_plan_inventory(sim_number, plan):
    if sim_number:
        # SIM - 
        sim_inventory = get_inventory_by_itemtext1(session,
sim_number, None)
        if not sim_inventory:
            # 
            service = get_inactive_service_by_sim(sim_number)
            if service:
                sim_inventory =
get_sim_for_service(service.service_id)
            else:
                # SIM
                sim_inventory =
get_first_available_inventory_by_item_name(
                    session,
                    "SIM Card",
                    filters={"item_state": ["New"], "customer_id": [None]}
                )

        # ICCIDMSISDN
        iccid = sim_inventory.itemtext1

        # 
        number_inventory = get_first_available_inventory_by_item_name(
            session,
            "Mobile Number"
        )
        msisdn = number_inventory.itemtext1

    return {
        "SIM Card": sim_inventory.inventory_id,
        "Mobile Number": number_inventory.inventory_id
    }

```

4.

```

def _post_signup_tasks(customer_id, plan, payment_method):
    # 1. 保存支付方式
    save_stripe_payment_method(customer_id, payment_method)

    # 2. 给顾客充值
    charge_customer_card(customer_id, plan.retail_setup_cost)

    # 3. 生成JWT令牌
    tokens = login_customer(customer_id)

    # 4. 初始化SIM卡库存
    inventory_vars = _populate_base_plan_inventory(sim_number,
plan)

    # 5. 初始化配置变量
    provision_vars = {
        "product_id": plan.product_id,
        "customer_id": customer_id,
        **inventory_vars # 添加SIM卡库存ID
    }

    # 如果有附加链
    if plan.addon_chain:
        provision_vars["addon_chain"] = plan.addon_chain

    provision_id = create_provisioning_job(provision_vars)

    # 6. 返回数据
    return {
        "access_token": tokens.access_token,
        "refresh_token": tokens.refresh_token,
        "customer_id": customer_id,
        "provision_id": provision_id
    }

```

5. 初始化SIM卡库存

初始化SIM卡库存

- **SIM**卡库存初始化
- **eSIM**卡库存初始化

6. JWT 토큰

JWT 토큰은 다음과 같습니다.

```
{
  "sub": "customer_123",
  "signup_process": true,
  "permissions": ["VIEW_OWN_PROVISION", "UPDATE_OWN_INVENTORY"]
}
```

이 토큰은

- 사용자 정보/권한
- 로그인 여부
- 사용자 권한 (SIM 등)

이 토큰은

다음과

- SIM 정보
- SIM 번호
- SIM 정보 (IMEI, ICCID, etc.)
- 사용자 SIM 정보
- 사용자 정보

이 토큰은

- 사용자 정보
- 사용자 권한
- 사용자 정보

Ansible

이 SIM 정보는 Ansible을 사용하여 관리할 수 있습니다.

Ansible/Ansible

SIM

/0mniCRM-API/Provisioners/plays/play_psim_only.yaml

SIMESIM

-
-
- IMS

1,561

Ansible -

```
- name: OmniCore Service Provisioning 2024
hosts: localhost
gather_facts: no
become: False

tasks:
  - name: Main provisioning block
    block:
      # --- PRE-PROVISIONING (Lines 240-396) ---
      - name: Get SIM information from inventory
        # Retrieves IMSI, ICCID, etc.

      - name: Validate IMSI length
        # Must be exactly 15 digits

      - name: Check if IMSI exists in HSS
        # Prevents duplicate provisioning

      - name: Get AuC ID for IMSI
        # Retrieves auc_id reference from HSS

      # --- HSS PROVISIONING (Lines 398-534) ---
      - name: Create/Update HSS Subscriber
        # Provisions to all HSS peers

      - name: Create IMS Subscriber
        # Enables voice calling

      # --- OCS SETUP (Lines 536-837) ---
      - name: Create ENUM entry
        # E.164 number mapping

      - name: Create FilterS rule
        # Account identification

      - name: Create AttributeS profile
        # IMSI/MSISDN mapping

      - name: Create ResourceS profile
        # Concurrent session limits

      - name: Create OCS account
        # Billing account creation
```

```

# --- SERVICE & INVENTORY (Lines 865-946) ---
- name: Create service record
  # In CRM database

- name: Assign SIM to service
  # Update inventory

- name: Send welcome SMS
  # Customer notification

rescue:
  # --- CLEANUP/DEPROVISIONING (Lines 975-1562) ---
  - name: Return inventory to pool
    # Reset customer_id, service_id to null

  - name: Delete OCS account
    # Remove billing account

  - name: Remove HSS subscribers
    # Or set to dormant state

  - name: Determine success/failure
    assert:
      that:
        - action == "deprovision"

```

□□□□□□

1. □□□□□

- **name:** Get SIM information from CRM inventory
uri:
url: "{{ crm_config.crm.base_url
 }}/crm/inventory/inventory_id/{{ inventory_id_sim_card }}"
method: GET
headers:
Authorization: "Bearer {{ access_token }}"
register: api_response_sim
- **name:** Extract and validate IMSI
set_fact:
imsi: "{{ api_response_sim.json.itemtext2 }}"
- **name:** Validate IMSI format
assert:
that:
 - imsi | length == 15
 - imsi is match('^[0-9]+\$')**fail_msg:** "IMSI must be exactly 15 digits"

2. HSS AuC ID

- **name:** Get AuC ID for IMSI from HSS
uri:
url: "{{ item }}/auc/imsi/{{ imsi }}"
method: GET
loop: "{{ crm_config.hss.hss_peers }}"
register: auc_lookup
- **name:** Extract auc_id from first HSS response
set_fact:
auc_id: "{{ auc_lookup.results[0].json.auc_id }}"

3. HSS

```

- name: Create subscriber in each HSS peer
  uri:
    url: "{{ item }}/subscriber/"
    method: PUT
    body_format: json
    body:
      enabled: true
      roaming_enabled: true
      auc_id: "{{ auc_id }}"
      msisdn: "{{ phone_number }}"
      imsi: "{{ imsi }}"
      ue_ambr_dl: 9999999 # 
      ue_ambr_ul: 9999999 # 
      apn_list: "1,2,3"
      default_apn: 1
      status_code: [200, 201]
    loop: "{{ crm_config.hss.hss_peers }}"
  register: hss_responses

```

4. IMS

```
- name: Create IMS subscriber for voice services
uri:
  url: "{{ item }}/ims_subscriber/"
  method: PUT
  body_format: json
  body:
    imsi: "{{ imsi }}"
    msisdn: "{{ phone_number }}"
    msisdn_list: "{{ phone_number }}"
    ifc_path: "default_ifc.xml"
    sh_profile: |
      <?xml version="1.0"?>
      <IMSSubscription>
        <PrivateID>{{ imsi }}@ims.mnc{{ mnc }}.mcc{{ mcc
        }}.3gppnetwork.org</PrivateID>
        <ServiceProfile>
          <PublicIdentity>
            <Identity>sip:{{ phone_number }}@ims.mnc{{ mnc
            }}.mcc{{ mcc }}.3gppnetwork.org</Identity>
          </PublicIdentity>
        </ServiceProfile>
      </IMSSubscription>
    status_code: [200, 201]
  loop: "{{ crm_config.hss.hss_peers }}"
```

5. OCS

创建IMSI/MSISDN

- name: Create FilterS for IMSI identification

uri:

url: "{{ crm_config.ocs.ocsApi }}/jsonrpc"

method: POST

body_format: json

body:

method: "ApierV1.SetFilter"

params:

- ID: "FILTER_IMSI_{{ service_uuid }}"
Type: "*string"
Element: "~*req.OriginHost"
Values: ["{{ imsi }}"]

创建IMSI

- name: Create AttributeS profile

uri:

url: "{{ crm_config.ocs.ocsApi }}/jsonrpc"

method: POST

body_format: json

body:

method: "ApierV1.SetAttributeProfile"

params:

- ID: "ATTR_ACCOUNT_{{ service_uuid }}"
FilterIDs: ["FILTER_IMSI_{{ service_uuid }}"]
Attributes:
 - Path: "*req.Account"
Value: "{{ service_uuid }}"

创建OCS

- name: Create OCS account

uri:

url: "{{ crm_config.ocs.ocsApi }}/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

初始化

- name: Add initial monetary balance
 - uri:
 - url: "{{ crm_config.ocs.ocsApi }}/jsonrpc"
 - method: POST
 - body_format: json
 - body:
 - method: "ApierV1.AddBalance"
 - params:
 - Tenant: "{{ crm_config.ocs.ocsTenant }}"
 - Account: "{{ service_uuid }}"
 - BalanceType: "*monetary"
 - Balance:
 - Value: 0
 - ExpiryTime: "+8760h" # 1月

6. 初始化

- name: Create service record in CRM
 - uri:
 - url: "{{ crm_config.crm.base_url }}/crm/service/"
 - method: PUT
 - headers:
 - Authorization: "Bearer {{ access_token }}"
 - body_format: json
 - body:
 - customer_id: "{{ customer_id }}"
 - product_id: "{{ product_id }}"
 - service_name: "Mobile Service - {{ phone_number }}"
 - service_uuid: "{{ service_uuid }}"
 - service_status: "Active"
 - retail_cost: "{{ monthly_cost }}"
 - status_code: 200
 - register: service_creation_response
- name: Extract service_id
 - set_fact:
 - service_id: "{{ service_creation_response.json.service_id }}"

7. 测试

```
- name: Assign SIM card to customer and service
  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 }}"
    service_id: "{{ service_id }}"
    item_state: "Assigned"
    sold_date: "{{ ansible_date_time.iso8601 }}"

- name: Assign phone number to customer and service
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ inventory_id_phone_number }}"
  method: PATCH
  headers:
    Authorization: "Bearer {{ access_token }}"
  body_format: json
  body:
    customer_id: "{{ customer_id }}"
    service_id: "{{ service_id }}"
    item_state: "Assigned"
    sold_date: "{{ ansible_date_time.iso8601 }}"
```

测试脚本

测试脚本在Ansible rescue中

测试脚本在Ansible rescue中 - 测试脚本在Ansible rescue中

```

rescue:
  # []
  # 1. []
  # 2. action == "deprovision"[]

- name: Get inventory items linked to service
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/customer_id/{{ customer_id }}"
  method: GET
  headers:
    Authorization: "Bearer {{ access_token }}"
  register: inventory_items
  ignore_errors: true

- name: Return inventory to available pool
  uri:
    url: "{{ crm_config.crm.base_url
  }}/crm/inventory/inventory_id/{{ item.inventory_id }}"
  method: PATCH
  headers:
    Authorization: "Bearer {{ access_token }}"
  body_format: json
  body:
    customer_id: null
    service_id: null
    item_state: "Used"
  loop: "{{ inventory_items.json.data }}"
  when: inventory_items.json.data is defined
  ignore_errors: true

- name: Delete OCS account
  uri:
    url: "{{ crm_config.ocs.ocsApi }}/jsonrpc"
    method: POST
    body_format: json
    body:
      method: "ApierV1.RemoveAccount"
      params:
        - Tenant: "{{ crm_config.ocs.ocsTenant }}"
          Account: "{{ service_uuid }}"
  ignore_errors: true

```

```

- name: Delete or disable HSS subscriber
  uri:
    url: "{{ item.key }}/subscriber/{{ item.value.subscriber_id
  }}"
    method: DELETE
    loop: "{{ hss_subscriber_data | dict2items }}"
    when: deprovision_subscriber | default(false) | bool
    ignore_errors: true

# 测试
- name: Determine if this was intentional deprovision or failed
  provision
  assert:
    that:
      - action == "deprovision"
    fail_msg: "Provisioning failed and rollback completed"
    success_msg: "Deprovisioning completed successfully"

```

测试

- `ignore_errors: true` 测试失败时不报错
- `action == "deprovision"` 测试失败时返回 0 = 成功
- `action` 测试失败时返回 2 = 失败
- 测试失败时返回 2 = 失败

测试SIM卡

测试本地SIM卡 `play_local_mobile_sim.yaml` - 测试HSS/OCS

MSISDN 测试 `play_swap_msisdn.yaml` - 测试HSS/OCS

测试/测试

- `play_topup_monetary.yaml` - 测试充值
- `play_topup_no_charge.yaml` - 测试充值/测试
- `play_topup_dongle.yaml` - 测试充值/测试

HSS & IMS

OmniCRM > HSS

HSS

URL: /OmniCRM-API/Provisioners/hss.py

Python

```
def ProvisionMobileSIM(json_data, hss_urls):
    """
    Provisions a mobile SIM to HSS and IMS

    Args:
        json_data: Dict with IMSI, MSISDN, authentication
        credentials
        hss_urls: List of HSS peer URLs

    Returns:
        Dict with subscriber IDs from each HSS peer
    """
    # Create HSS subscriber (for data)
    # Create IMS subscriber (for voice)
    # Return subscriber IDs for tracking

def get_subscriber_info_msisdn(msisdn, hss_urls):
    """
    Retrieves subscriber details by phone number

    Args:
        msisdn: Phone number to lookup
        hss_urls: List of HSS peer URLs

    Returns:
        Subscriber information from HSS
    """
```

HSS

OmniCRM HSS

```
# crm_config.yaml
hss:
  hss_peers:
    - "http://10.12.64.140:8080"
    - "http://10.12.64.141:8080"
  apn_list: "1,2,3"
  default_apn: 1
```

□□□□□□□□HSS□□□□□□□□

- 
- 
- 
- 

□□□□□AuC□

AuC[] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] auc_id[]

IMSIAuC ID

```
GET /auc/imsi/{imsi}
```

111

```
{
  "auc_id": 12345,
  "imsi": "310120123456789",
  "iccid": "8961234567890123456",
  "esim": false,
  "lpa": null
}
```

esim true lpa

Ki OPC PIN PUK HSS API auc_id

OCS

OCS OmniCRM CGRateS OCS

OCS

/OmniCRM-API/Provisioners/ocs.py

```
# crm_config.yaml
ocs:
  cgrates: "10.64.12.160:2080"
  ocsTenant: "mnc001.mcc001.3gppnetwork.org"
  ocsApi: "http://10.64.12.160:2080"
```

SIM OCS

1. -

```
{
  "ID": "FILTER_IMSI_Service_abc123",
  "Type": "*string",
  "Element": "~*req.OriginHost",
  "Values": ["310120123456789"]
}
```

2. -

IMSI

```
{
  "ID": "ATTR_ACCOUNT_Service_abc123",
  "FilterIDs": ["FILTER_IMSI_Service_abc123"],
  "Attributes": [
    {
      "Path": "*req.Account",
      "Value": "Service_abc123"
    },
    {
      "Path": "*req.Bandwidth",
      "Value": "100000000" // 100 Mbps
    }
  ]
}
```

3. 属性 - 限制

限制属性配置示例

```
{
  "ID": "RES_SESSIONS_Service_abc123",
  "FilterIDs": ["FILTER_IMSI_Service_abc123"],
  "Limit": 5, // 限制5个会话
  "UsageTTL": "-1"
}
```

4. 属性 - 统计

统计属性配置示例

```
{
  "ID": "STATS_Service_abc123",
  "FilterIDs": ["FILTER_IMSI_Service_abc123"],
  "Metrics": [
    "*sum#~*req.Usage",
    "*tcc" // 统计
  ]
}
```

5. 請求 - 応答

リクエスト/レスポンスの例

```
{
  "Id": "ActionPlan_Service_abc123_Monthly",
  "ActionsId": "Action_Add_Monthly_Data",
  "Timing": {
    "MonthDays": [1], // 1日
    "Time": "00:00:00Z"
  }
}
```

OCS API

ocs.pyの例

```
async def async_get_balance(account, server, tenant):
    """取得"""

async def async_topup_dongle(tenant, account, server, days):
    """リチャージ"""

def Get_Account_Status(account, server, tenant):
    """ステータス取得"""

def CGRateS_API_Call(method, params, server, tenant):
    """CGRateS JSON-RPC API"""
```

SIM

OmniCRMとCRM/HSSの連携

パス

例: /OmniCRM-API/Provisioners/hss_reconcile.py

目標

- HSSからSIM情報を取得
- SIM情報をDBに保存
- MSISDN取得
- HTML出力

前提条件

1. HSSからSIM情報取得

- SIM情報を取得
- IMSIからHSSに接続
- HSSからSIM情報を取得

2. SIM情報をDBに保存

- HSSから取得したSIM情報をDBに保存
- IMSIからHSSに接続
- HSSからSIM情報を取得

3. MSISDN取得

- SIM情報をDBから取得
- SIM情報をDBから取得
- SIM情報をDBから取得

実行方法

```
python hss_reconcile.py
```

結果

- HTML出力
- DBに保存
- 実行

備考

- 物理SIM
- 虚拟SIM
- 物理SIM卡
- 虚拟SIM卡

SIM卡管理

物理SIM卡

物理SIM卡管理“IMSI管理HSS”

- 物理IMSI管理
- 物理HSS管理
 - 物理HSS管理
 - 物理HSS管理
 - 物理HSS管理

物理eSIM管理

- 物理management_url管理LPA
- 物理eSIM管理management_url
- 物理LPA:1\$server\$activation-code

物理SIM卡管理“物理”管理

- 物理SIM卡管理
- 物理SIM卡管理
- 物理SIM卡管理

物理SIM卡管理

- 物理IMS管理
- 物理IMS管理
- 物理IMS管理

物理SIM卡管理SIM卡

- 通过 SIM 向 HSS 注册 Ki/OPC
- 通过 AuC 向 HSS 注册 SIM
- 通过 HSS 向 AuC 注册

通过 SIM 注册

- 通过 HSS 注册 SIM
- 通过 HSS 注册 SIM 时，customer_id 为 NULL
- 通过 HSS 注册 SIM 时，SIM 类型为“普通”

通过 HSS 注册

1. 通过 HSS 注册

```
GET /crm/provision/provision_id/{id}
```

通过 HSS 注册 2 个 SIM 卡

2. 通过 HSS 注册

```
GET {hss_url}/subscriber/msisdn/{phone_number}
```

通过 HSS 注册 2 个 SIM 卡

3. 通过 OCS 注册

```
{
  "method": "ApierV2.GetAccount",
  "params": [
    {
      "Tenant": "mnc001.mcc001.3gppnetwork.org",
      "Account": "Service_abc123"
    }
  ]
}
```

4. 通过 OCS 注册

GET /crm/inventory/?filters={"itemtext2":["310120123456789"]}

IMSI

5. Ansible

/var/log/ansible/

SIM

- 1. SIM
- 2. HSS/
- 3.
- 4. SIM
- 5. /

- 1. Ki/OPC HSS CRM
- 2. /SIM
- 3. SIM
- 4. PIN/PUK PIN/PUK HSS
- 5. / HSS

- 1. IMSI
- 2. /
- 3.
- 4.
- 5.

[illegible][illegible][illegible][illegible]

[illegible]

A horizontal number line with 100 small squares representing units. The squares are numbered from 0 to 100. The square for the number 25 is highlighted in orange.

[illegible][illegible][illegible]

IP

服務條款

本服務條款係根據中華民國消費者保護法及相關法規制定，凡使用本服務之用戶，均須同意本條款之內容。

一、總則

1. 服務範圍

1. 網路服務
2. 數位內容服務
3. 網路通訊服務SIM卡服務
4. 網路廣播電視服務IPTV服務
5. 網路電話服務VoIP
6. 其他相關服務

本服務條款之內容，係根據中華民國消費者保護法及相關法規制定，凡使用本服務之用戶，均須同意本條款之內容。

二、用戶之權利與義務

1. 用戶之權利

2. 用戶之義務

3. 用戶之權利

4. 用戶之義務

三、其他事項

1. 隱私權政策

本服務條款之內容，係根據中華民國消費者保護法及相關法規制定，凡使用本服務之用戶，均須同意本條款之內容。

2. 数据概览

系统记录的所有用户活动数据概览

数据源

用户列表 [John ▼]

系统记录 • John Smith ID 123 • John Doe ID 456 • Johnson Enterprises ID 789

数据项

- 用户ID
- 用户姓名
- 用户邮箱
- 用户角色

数据更新时间



警告

系统检测到异常活动，请立即检查用户权限设置。

3. 数据操作

系统记录的所有用户活动数据概览 `inventory_items_list`

Page 10 of 10

SIM ☐ *

□□□□ *

□□□□□□□

-

- □□□□□□□□□□□□□□

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

4

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

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □

[] []

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

- 
-

[illegible]

5

[illegible]

John Smith ID 123 - 20GB

• SIM-SIM-00123-ICCID-8944...0001 • +44 7700 900123

□□□ • □□□□□£0.00 • □□□□£15.00

□ □ □ □ □ □

[] []

□□**"□□□□□"□□□□□□□□□□

6

[illegible]

✓ 已验证 ✓ SIM卡ICCID 8944...0001 ✓ 手机号+44 7700 900123 ☎
OCS 已验证... 已验证 已验证

6 3

0.2 API

☒ 已验证

☎ 已验证

已验证

☒ 已验证

7

John Smith

ID 789 - +44 7700 900123

[] []

**" " **

OCS

ID 456

[] [] []

-
-
-

</

[] []

□ □ □ □ □ □

□ □

1. CSVファイル
2. テキストファイル
3. 圧縮ファイル
4. 二進制ファイル

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

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

步骤2 绑定设备

1. 绑定设备名称 → **John Smith**
2. 绑定设备
3. 输入**“设备名称”**
4. 绑定设备
5. 输入**“设备 - 100Mbps”**
6. 绑定设备名称/设备
7. 绑定设备CPE设备
8. 绑定设备❖❖设备
9. 绑定
10. 绑定设备

步骤3 绑定设备

1. 绑定设备“绑定设备”
2. 绑定设备名称
3. 绑定设备名称
4. 绑定设备名称“绑定设备”
5. 输入“5GB设备”
6. 绑定
7. 绑定设备
8. 绑定设备名称

绑定设备

“绑定设备”

- 绑定 绑定设备名称
- 绑定
 - 绑定设备名称
 - 绑定设备名称

- 國際標準化組織

“認證”

- 認證 認證過程
- 認證
 - 認證過程
 - 認證過程“”“”
 - 認證過程認證過程

“認證”

- 認證 認證過程
- 認證
 - 認證
 - 認證ID認證
 - 認證

認證

- 認證 認證過程
- 認證
 - 認證5
 - 認證
 - 認證OCS
 - 認證

認證認證

- 認證 認證OCS
- 認證
 - 認證OCS
 - 認證SIM/HLR/HSS
 - 認證RADIUS/DPI
 - 認證

□□□□

□□□□□

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

□□□□□

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

□□□□□

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

□□□□□□

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

□□□□

- `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

□□□□ (CDRs)

00000000 0000000000 00000 00000 00000000000000000000 //00000
000000000000000000

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

□□□□□□ □□□□□□ □□□□□□□□□□□□□□□□

□ □ □ □ □

1. 氨基酸序列 氨基酸 氨基酸序列
2. 可变区 (CDRs) 氨基酸

CDRs

--	--	--	--

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

□	□□□□
□□□□	□□□□□□□□□□□□
□□□	□□□□□□□□□□□□□□  □□
□□	□□□ □□□□□ □ □□ □□
□□□□	□□□□ (HH:MM:SS)□□□□ (MB/GB) □□□□□
□□	□□□□□□□□

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

[illegible]

□□□□

□□□□□□□□ □□□□□ □□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□ □□ □□□□□□

□□□□□□

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

- 
- 

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

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

- 5G - 5G 网络
- 4G - 4G 网络
- WiFi 6 - WiFi 6 网络
- IP - IP 地址

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

□ □ □ □ □ □ □ □

1. 
2. “” “”
3.

□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- 00000 - 00000000000000000000000000000000 VoIP 00
- 00000 - 0000000000000000
- 000000 - 0000000000000000
- 0000 - 00000000

□ □

□ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □

Modem-12345 - TP-Link AX1800 • Modem-12346 - TP-Link AX1800 • Modem-12347 - Netgear RAX40

3

[] []

-
-

4

“”

✓ ✓ ↺ ...

-
- Ansible provisioning_play
-
-
- OCS
-

-

項目別見積書

お客様のご要望に基づき、以下のサービスを提供いたします。

サービス料

5GB 月額

2025年1月10日 から 2025年1月17日 までの期間 月額 £5.00

WiFi 6 対応機器

2024年12月1日 から Modem-12345 までの期間 月額 £10.00

作業内容

作業内容の詳細

- 現場調査
- “作業内容” の確認
- 作業内容 “” の確認
- 作業

作業時間

- 作業時間 - 作業内容
- 作業時間 - 作業内容
- 作業時間 - 作業内容
- 作業時間 - 作業内容



作業内容 CPE 作業内容

作業内容

作業 1

- 作業内容 “作業内容”

- 管理する製品とサービス
- 管理する製品とサービス

管理する製品とサービス

- 管理する製品とサービス
- 管理する製品とサービス
 - 管理する製品とサービス
 - 管理する製品とサービス
 - 管理する製品とサービス `inventory_items_list` 管理

管理する製品とサービス

- 管理する製品とサービス
- 管理する製品とサービス
 - 管理する製品とサービス `auto_renew: true`
 - 管理する製品とサービス
 - 管理する製品とサービス

管理する製品とサービス

- `guide_product_lifecycle` - 管理する製品とサービス
- `csa_add_service` - 管理する製品とサービス
- `concepts_products_and_services` - 管理する製品とサービス
- `administration_inventory` - 管理する製品とサービス

[illegible]

1111

1111

□ □



□ □ □ □ □ □ □ □ □ □ □ □ □

11

- **ID** - 00000
- 0000 - 0000000000 "00 - +44 7700 900123"

- 網路 - 網路網路網路網路
- 網路 - 網路IPTV網路網路VoIP
- 網路 - 網路網路網路網路
- 網路 - 網路網路
- 網路 - 網路/網路
- 網路 - 網路
- 網路 - 網路網路網路網路網路網路網路網路網路網路??

網路網路

網路網路

網路網路網路網路網路網路

- 網路 - 網路網路網路網路網路
- 網路 - 網路網路網路網路網路
- 網路 - 網路網路網路網路
- 網路 - 網路網路網路網路網路網路網路網路

網路網路網路網路網路網路網路網路網路網路

網路網路網路網路

網路網路網路

網路網路網路網路網路網路網路網路

網路

- 網路 - 網路網路 SIM 網路
- 網路 - 網路網路DSL 網路網路
- **IPTV** - 網路網路網路
- **VoIP** - 網路網路網路

00000

000000000

0000000000000000

00000000000000

000 00000000000000

00

0000000000000000

00000

- 0000
- 00 ID
- 0000
- 00000000000000
- 00 UUID

000 00“0770”0000000000“0770”0000

0000

0000000000000000

00000

- 00 ID0000000000
- 00000000000000
- 00000000000000
- 00000000000000
- 00000000000000
- 00000000000000
- 0000000000

□□□□

- □□□□□□□□A-Z□□□-□□□□□-□□□
- □□□□□□□□Z-A□□□-□□□□□-□□□
- □□□□□□□ ▲ □ ▼ □□□

□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□

1. □□ - □□□□□□□□□□□□□□
2. □□ - □□□□□□SIM □□□□□□□□□□
3. □□ - □□□□□□□□□
4. □□□□ - □□/□□/□□□□□□□□□□
5. □□□□ - □□□□□□□□□□□□
6. □□□□ - □□□□□

□□□□□

□□□□□□□□□□□□□□□□ : □□□□□□□□□

- □□ - □□□□□□□
- □□ - □□□□□□□
- □□□□ - □□□□□□□□□□□□□□
- □□□□ - □□□□□□□□□□□□□□
- □□ - □□/□□□□□

□□□□

□□□□□□□□□□□□□□□□

□□□□

□□□□

□□□□□□□□□□□□□□□

□□

□□□□□□□□□□□□□□□□

| ☒ □□ 1 ☒ □□ 2 ☒ □□ 3

□□□□□□

□□□□□□□□□□□□

□□□□

□□□□□□□□□□□□□□

□□□□□□

1. □□□□□□□□
2. □□ “□□□□” □□
3. □□□□□□□□

 □□

□□□□□□□□□□□□

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

111

- 
- 
- 

[illegible]

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

□ □ □ □

□ □ □ □ □ □ □ □ □ □

11

- 
- 
- 
- 

□□□

- 00000000
- 0000
- 00000000
- 0000000
- 00000000/00

11

- □□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□□/□□□□□
- □□□□□□□□□□□□□□□□□□□□□□

□□□□□□

□□□□□□□

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

□□□□□□

- □□ → □□□□ □□□□□□□□□□□□
- □□ → □□□ □ OCS/□□□□□□□□□□□□
- □□ → □□□ □□□□□□□□□□
- □□□ → □□□ □□□□□□□□□□□□

□□□□□□

□□□□□□□□ “□□□□” □□□□□□□□

□□□□□□

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

- 0000 - 0000/00000000

111

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

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

[illegible]

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

□ □ □ □

□□□□

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

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

11

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

□□□□□

API 仕様

本ドキュメントは、CGRateS の API 仕様を説明します。

1. 概要 - CGRateS の基本情報と ActionPlan
2. 認証 - API へのアクセス方法
3. エラー - エラーメッセージの説明
4. 詳細 - 各 API エンドポイントの詳細

概要

CGRateS は、API を提供し、CRM システムと連携します。API の詳細は、API ドキュメントを参照してください。

認証

API にアクセスするには、認証が必要です。

1. API キー (API Key) を取得
2. API キーを使用して認証

例: Acme Corp の FixedWireless_75628fa5 の Mega の料金は \$89.99

2025年11月11日 10:45

[API] [詳細]

3. API “詳細” を取得

4. 消息接收器

5. 消息接收器

消息接收器

- 消息接收器
- CGRateS 消息接收器
- 消息接收器
- 消息接收器
- 消息接收器

消息

- 消息接收器
- 消息接收器
- 消息接收器
- 消息接收器

消息接收器

消息接收器

1. 消息接收器 (消息接收器) 消息

2. 消息接收器

消息接收器

消息Acme Corp 消息FixedWireless_75628fa5 消息 2025-11-11 消息
10:45

消息接收器

[消息] [消息接收器]

3. 消息 “消息接收器” 消息

4. ActionPlan 消息 CGRateS 消息

5. 設定するパラメータ

パラメータ

- ActionPlan の CGRateS 設定
- 設定値
- 設定値の単位
- 設定値の範囲

設定

- 設定値
- 設定値の単位
- 設定値の範囲
- 設定値の範囲



設定値の範囲

設定値

設定

- 設定値
- 設定値
- 設定値 “N/A”

設定

- 設定値/UUID
- 設定値
- 設定値 UUID
- 設定値 “N/A”

設定

- CGRateS ActionPlan 設定

- 0000000000000000
- 000000000000000000000000

0000

- 00 CRM 0000000000
- 0000000000
- 000000000000000000000000 “N/A”

000000

- 0000000000000000
- 0000000000000000 “3000” “200”
- 0000000000000000 CGRateS 00000000

0000

- 00 CRM 000000000000
- 000000 - 00000000
- 000000 - 00000000
- 000000 - 000000000000000000000000

0000

000000/00000000 “N/A”

- 0000 CGRateS 000000 UUID 0 CRM 000000 service_uuid 0000
- 0000
 - 000000000000000000000000
 - 00 service_uuid 0000000000000000 ServiceType_UUID 000000000000000000000000 FixedWireless_75628fa5 0
 - 00000000 CRM 0000000000 CGRateS 0000

0000 “N/A”

- 0000 CGRateS 000000 ID 0 CRM 0000000000
- 0000
 - 000000000000000000000000

- 在 ActionPlan 中 product_id 在 CRM 中
- 在

“”

- 在 CGRateS 中 NextRunTime
- 在
 - 在 CGRateS ActionPlan 中
 - 在 ActionTiming 中
 - 在 CGRateS 中

“”

- 在 中
- 在
 - 在
 - 在 Ansible playbooks 中
 - 在 OCS 中
 - 在

- 在 中 CGRateS 在
- 在
 - 在 “” 中
 - 在 ActionPlan 中 CGRateS 在
 - 在 CGRateS API 中

◀ 1 [2] 3 4 5 ▶

[10 ▼]

- 10

CGRateS `cgates_api_access` `rbac`

□ □ □ □ □ □ □ □ □

CGRateS Ansible playbooks

- CGRateS 00000000
- 000000IMSI00MSISDN00000000000000000000
- 0000000000000000
- 000000000000

□ □

- `MaxBitrateDL/UL`
- `PcefPolicyName`
- `MSISDN`
- `QoS`
- `MSISDN`
- `QoS`

[illegible]

- 0000 - 0000000000000000
- 01 - 00000000
- 0001 - 00000000000000
- 000001 - 000000000000000000000000



CGRateS

CGRateS

CGRateS

- **CGRateS** ☐ - ☐

- **CGRateS** 関数 - 関数呼び出し
- 関数 - 関数呼び出し

関数呼び出し関数呼び出し関数呼び出し関数呼び出し/関数呼び出し

XX

CGRateS 簡介

XX

用戶 ID 對應 ATTR_ACCOUNT_{service_uuid}

用戶 用戶 UUID 對應 Mobile_SIM_c2880638 對應XXXXXXXXXXXX ID 對應
ATTR_ACCOUNT_Mobile_SIM_c2880638

XXXXXXXXXX

1. XXXXXXXXXX
2. 用戶 “CGRateS 用戶” 用戶
3. 用戶 “XXXXXX” 用戶
4. XXXXXXXX/XX/XXXXXX
5. 用戶 “XXXXXX”

□□□□□

- □□ - □□□□□□□□□□*req.Account□*req.IMSI□
- □□ - □□□□□□□□

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 - 測試環境測試環境 ~*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]

□ □ □ □

[illegible]

□ □ □ □ □ □ □

5. 如何“管理”项目成员？

□□□□□□□□

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

□□□□□□□□

□□“□□□□”□□□□□□□□□□□□

- 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

000000

00000000000000000000“00”000000000000

 00

0000000000000000 UUID000000000000 `*string:~*req.Account:{service_uuid}`
00000000

CGRateS API 00

00 CGRateS 00000000000000000000 OmniCRM API 000000

0000 `POST /crm/ocs/proxy`

000000

- `method` - CGRateS API 000000 `APIerSv1.GetAttributeProfile`
- `params` - 00000000
- `customer_id` - 00 ID0000000000
- `service_id` - 00 ID0000000000

000000

- `tenant` - CGRateS 000000000000

000000

```
{
  "method": "APIerSv1.GetAttributeProfile",
  "params": [{"ID": "ATTR_ACCOUNT_Mobile_SIM_c2880638"}],
  "customer_id": 123,
  "service_id": 456
}
```

□□□□□

□□□□□ OmniCRM □□□□ (crm_config.yaml) □□ ocs.ocsTenant □□□□□□□
CGRateS □□□□□□□□□□□□□□□□□□

□□□□□

□□ cgrates_api_access □□□□□□□□□□ admin □□□

□□□□□

□□□ GET □ CGRateS API □□□□□□□□□□□□□□□□□□

- □□□ API □□
- □□□□□
- □□□□□□□
- □□□□□□□ ID
- □□□□□□□
- □□□

□□□□□ CGRateS □□□□□□□□□□□□

□□□□ **CGRateS** □□

□□ **CGRateS** □□□□□“□□□□□□”

- □□□ □□□□ cgrates_api_access □□
- □□□ □□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□

- □□□ CGRateS □□□□□□□□□□□□□□
- □□□
 - □□□□□□□ CGRateS □□□□□□□
 - □□□□□□□□□□□
 - □□□□□□□□□□□ API □□
 - □□□□□□□□□□□□□□□□□□

目標

- 目標 1: CGRateS API 利用
- 目標 2
 - 目標 2-1: 目標 2-1-1
 - 目標 2-2: JSON 形式で取得
 - 目標 2-3: 目標 2-3-1
 - 目標 2-4: CGRateS 利用

前提条件

- 前提条件 1: 目標 1-1
- 前提条件 2
 - 前提条件 2-1: 目標 2-1-1
 - 前提条件 2-2: 目標 2-2-1
 - 前提条件 2-3: UUID 形式で CGRateS 利用

目標 1

- 目標 1-1: 目標 1-1-1
- 目標 1-2: 目標 1-2-1

目標 2: CGRateS 利用

- 目標 2-1: GET 形式で取得
- 目標 2-2: 目標 2-2-1 - 目標 2-2-2

目標 3

目標 4: 1. 目標 4-1

1. 目標 4-1-1
2. 目標 4-1-2
3. 目標 4-1-3
4. 目標 4-1-4
5. 目標 4-1-5

問題 2 読者の理解を深める

1. 本文 “ ” の
2. “ ” の
3. の
4. の

問題 3 読者の理解を深める

1. の “ ”
2. の
3. の
4. “ ”
5. の

問題 4 読者の理解を深める

1. の
2. の
3. の
4. “ ”
5. の

問題 5 読者の理解を深める

1. の の
2. の
3. “ ” の
4. / / SMS の
5. の

- 0000 000000000000
- 0000 00“0000”0000000000

0000000000

- 0000 000000000000
- 0000
 - 0000000000“0770”00000000
 - 00000000
 - 00000000

0000000000

- 0000 00000000
- 0000 000000000000000000

00000

- 0000 000000
- 0000 0000 ▲▼ 00000000

0000000

- 0000 0000000000
- 0000
 - 00000000000000
 - 0000000000
 - 0000000000

000000

- `csa_add_service` - 000000
- `csa_modify` - 000000000000
- `concepts_products_and_services` - 00000000
- `basics_customers` - 00000

ОБЩЕЕ ПОЛОЖЕНИЕ

Настоящее Положение разработано в соответствии с требованиями Федерального закона от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации» (ОЗНИТИ) и ОКС (Общественная Контрастная Система) для обеспечения безопасности информации.

Цели

Цели настоящего Положения:

- Обеспечение безопасности информации.
- Обеспечение конфиденциальности информации.
- Обеспечение целостности информации.
- Обеспечение доступности информации.
- Обеспечение актуальности информации.

Область применения

Область применения настоящего Положения:

1. Все информационные ресурсы.
2. Все информационные ресурсы (:)
3. Все "информационные"

Область применения:

1. Все информационные ресурсы
2. Все "информационные" ресурсы

Область применения:

1. Все информационные ресурсы
2. Все информационные ресурсы
3. Все информационные ресурсы "информационные"

RAM 12.5GB / 20GB (62%)

Specifications

- Processor - 5GB/10GB/20GB options
- Storage - SSD options 500GB/1TB
- OS - Windows 10/11
- **USB** ports - USB-C/Thunderbolt

Features

Key Features

- High performance
- Scalable architecture
- Advanced security features

Performance

System Performance

Test Results (min)

Details

Time 125 min / 500 min (25%)

Configuration

- Processor
- Memory
- Storage
- Network

Notes

For more information, please refer to the user manual.

□□□□

□□□□□□□□

□□□ □□ (msgs)

□□□□□

| □□□45 / 250 (18%)

□□□□□

- □□□□□160 □□□□
- □□□□□□□□□
- □□□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□

□□□ □□ (£, \$, €, □)

□□□□□

| □□□£15.50 / £20.00 (77%)

□□□□□

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

□□□□□□□□

□□□□□□□□ "□□□□" □□□□□□□□□□□□□□

□□□□□

□ □□□ 20GB

□□□12.5 GB □□□7.5 GB □□□2025□1□25□□15□□□□ 10

□ □□□ 5GB

□□□5.0 GB □□□0 GB □□□2025□1□31□□21□□□□20

□ □□□ 3GB

□□□0 GB □□□3.0 GB □□□2025□1□18□□□□□□□□30

□□□□17.5 GB

□□□□□□

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

□□□□□

□□□□□□□□□□□□□□□□

- □□□□ = □□□□
- □□□□ = □□□□

□□□□□□

- □□□□□□□□ 30□□□□□□□□□□□□
- □□□□□□□□ 20□□□□□□□□
- □□□□□□□□ 10□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ CGRateS □□□□□□□□□□□□□□□□□□□□ CGRateS □□□□□□□□□□

-
-
-
-

-
-
-
-

-
-
-

- (GB)
- ()
- ()
- ()

12.5GB 125 min 45 msgs £15.50

/

- (GB TB)
-

450GB / 500GB £45.00

/ **USB**

USB

-
-

`dongle=true`

0 / 0

- OCS CGRateS
- OCS
 - OCS API
 - UUID

- OCS

- 詳細情報
 - 詳細情報
 - 詳細情報取得API
 - 詳細 OCS API 詳細情報

詳細情報

- 詳細 OCS 詳細情報取得API
- 詳細情報
 - 詳細 OCS 詳細情報
 - 詳細情報取得API
 - 詳細 OCS 詳細情報

詳細情報

- 詳細 詳細情報取得API
- 詳細情報
 - 詳細情報取得API
 - 詳細 OCS 詳細情報

詳細情報

- 詳細 詳細情報取得API
- 詳細情報
 - 詳細情報取得API
 - 詳細情報取得API
 - 詳細情報取得API

OCS/CGRateS

詳細情報 **OCS** 詳細情報取得API CGRateS 詳細情報取得API  詳細情報取得API
 詳細情報 **CGRateS** 詳細情報取得API

詳細情報

1. 詳細情報取得API
2. OmniCRM 詳細 `GET /crm/service/{service_id}`

3. API 返回 UUID 给 OCS

4. OCS 返回数据

```
{
  "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 显示数据 → GB → 元

6. 数据更新

7. 3 分钟

OCS 鉴权

OCS 鉴权是 OCS 的 UI 鉴权

- `*data` → 数据鉴权
- `*voice` → 语音鉴权
- `*sms` → SMS 鉴权
- `*monetary` → 货币鉴权

鉴权 = 鉴权 + 鉴权 + 鉴权

鉴权

- 鉴权 → GB 1024³
- 鉴权 → 鉴权 60×10⁹
- 鉴权
- 鉴权 → 鉴权 100

鉴权

鉴权 OCS 鉴权 CGRateS 鉴权

鉴权

鉴权 CGRateS 鉴权

- 鉴权
- 鉴权
- 鉴权
- 鉴权

鉴权

1. 鉴权

- 鉴权 `auto_renew = true`

- 查詢 OCS 查詢結果
- 查詢結果返回數據格式

2. 查詢結果

查詢結果

- 查詢 ID - 查詢 UUID
- 查詢 - 查詢結果返回數據格式
- 查詢 - 查詢結果返回數據 1 00:00 UTC
- 查詢 - 查詢結果

3. 查詢結果

- OCS 查詢結果
- 查詢結果返回數據 20GB 查詢 500 查詢
- 查詢結果返回數據 30 查詢
- 查詢結果返回數據

4. 查詢結果

- 查詢 OCS 查詢結果
- 查詢結果返回數據
- 查詢結果返回數據

查詢結果

```
{
  "ActionPlanId":
  "ProductID_MonthlyPlan__ProductName_20GB_Mobile__ActionPlan_Monthly_F
  "NextExecTime": "2025-02-01T00:00:00+00:00",
  "ActionName_hr": "查詢",
  "PlanName": "20GB 查詢",
  "ActionFrequency_hr": "查詢",
  "custom_NextExecTime_hr": "0 22 查詢"
}
```

查詢結果

- OCS
-

- CGRateS -
- CGRateS - PLMN
- features_topup_recharge -
- csa_service_management -
- csa_modify -
- concepts_products_and_services -

权限

权限管理

权限管理是系统安全的重要组成部分，它决定了用户能够访问哪些资源以及执行哪些操作。合理的权限管理可以有效防止数据泄露和系统被非法篡改。

权限管理通常包括用户管理、角色管理、权限分配和权限回收等功能。

权限管理模型主要分为 RBAC（基于角色的访问控制）和 ABAC（基于属性的访问控制）两种。

RBAC

RBAC 模型

1. 用户管理
2. 角色管理
3. 权限管理
4. 权限分配
5. 权限回收

ABAC

ABAC 模型

- can_impersonate 权限
- admin 权限

权限管理是系统安全的重要组成部分，它决定了用户能够访问哪些资源以及执行哪些操作。合理的权限管理可以有效防止数据泄露和系统被非法篡改。

权限管理

Web UI

1. 實現一個 CRM 系統
2. 實現一個系統
3. 實現“系統”系統
4. 實現系統
5. 實現系統

API

實現

POST /auth/impersonate

can_impersonate admin

實現

```
{  
  "user_id": 42  
}
```

實現

```
{
  "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"
}
```

API Response

API Request

API Parameters

- API Parameters
 - `impersonating_user_id` (required)
 - `target_user_id` (required)
 - `impersonation_start` (required)
- API Response
 - `impersonating_user_id` (required) ID
 - `target_user_id` (required)
 - `impersonation_start` (required)
- API Response
 - "Access Token" (required)
 - "Refresh Token" (required)
- API Response
 - `ImpersonationLog` (required)

API

Web UI

1. 認証画面からログイン
2. 認証画面から“ impersonate ”
3. 認証画面からログアウト

API

POST /auth/stop_impersonation

認証画面からログアウト

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"
}
```

認証画面からログアウト

API

認証画面からログアウト ImpersonationLog

- impersonating_user_id
- target_user_id
- impersonation_start
- impersonation_end
- impersonation_duration

認証画面からログアウト

- 权限管理
- 用户管理
- 角色管理
- 资源管理

权限管理

权限管理 API 接口

GET /auth/impersonation_logs?user_id={user_id}&start_date={date}&end_date={date}

权限管理

- 权限管理
- 用户管理
- 角色管理
- 资源管理

权限管理

1. 权限管理
2. 用户管理
3. 角色管理
4. 资源管理
5. 权限管理
6. 权限管理 `can_impersonate` 接口
7. 权限管理

权限管理

- 权限管理
- 用户管理
- 角色管理

- 認證碼驗證碼驗證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼驗證碼驗證碼驗證碼
- **MFA** 驗證碼 2FA 驗證碼驗證碼驗證碼

認證

- 認證碼驗證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼驗證碼驗證碼驗證碼

認證碼驗證碼

認證“認證碼”驗證碼

- 認證碼驗證碼 `can_impersonate` `admin` 認證
- 認證碼驗證碼驗證碼驗證碼驗證碼

認證碼驗證碼

- 認證碼  認證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼驗證碼驗證碼驗證碼

認證碼驗證碼

- 認證碼 API 認證碼 `POST /auth/stop_impersonation`
- 認證碼驗證碼 cookies 認證碼驗證碼

認證碼驗證碼

- 認證碼驗證碼驗證碼驗證碼驗證碼
- 認證碼驗證碼 - 認證碼驗證碼 ID

CGRateS

OmniCRM CGRateS

CGRateS CRM OCS Python

11

1. **CGRateS/OCS** - ██████████
2. **OmniCRM** - ██████████

Ansible Plays - CGRateS

CGRateS

CGRates   OmniCRM  CGRates  CGRates

- 國際間通訊服務
- 國際間通訊服務 MO 及 MT 簡訊/SMS 國際間通訊服務
- 國際間通訊2G/3G 簡訊 CAMEL、MO-MT SMS、USSD、LTE/5G 及 Diameter
- 國際間通訊服務國際間通訊服務
- 國際間通訊服務國際間通訊服務LCR國際間通訊服務

CRM - OmniCRM CGRateS CGRateS

API Test Script

Step 1: CRM API Authentication

```
import requests
import json

crm_url = 'https://crm.example.com/'
session = requests.Session()

headers = {
    "Content-Type": "application/json"
}

# Login
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 Authenticated")
```

Step 2: Data Retrieval

Retrieve data from the CRM system using the SIM card information.

```

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"✅✅✅✅✅{inventory_template['item']}")
else:
    # ❌❌❌❌
    response = session.put(
        crm_url + '/crm/inventory/item_template',
        json=inventory_template,
        headers=headers
    )
    assert response.status_code == 200
    print(f"✅✅✅✅✅{inventory_template['item']}")

```

❌❌❌❌❌

- **item**❌❌❌❌ - ❌❌❌❌
- **itemtext1_label** - ❌❌❌❌❌❌❌❌❌❌
- **itemtext2_label** - ❌❌❌❌❌❌❌❌❌❌
- **wholesale_cost** - ❌❌❌❌
- **retail_cost** - ❌❌❌❌❌❌
- **allow_dropdown_staff** - ❌❌❌❌❌❌❌❌❌❌
- **allow_dropdown_customer** - ❌❌❌❌❌❌❌❌❌❌
- **icon** - Font Awesome ❌❌❌

❌❌ 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 概要

本章は、CGRateS の概要について説明します。

概要

1. 対象 - 対象は、SMS 送信先として `Dest_AU_Mobile`、`Dest_International_UK`
2. **PLMN** 指定 - 対象は、PLMN 指定として `Dest_PLMN_OnNet`、`Dest_PLMN_US_Verizon`

対象

- **SMS** 送信先 → 対象は、SMS 送信先として
- **PLMN** 指定 → 対象は、PLMN 指定として

対象の PLMN

- 対象は、PLMN 指定として
- PLMN 指定として
- PLMN 指定として `mccXXX.mncYYY`
- 対象は、PLMN 指定として

対象の CGRateS 指定 - 対象は

第5章 CGRateS 詳細

本章は、CGRateS の詳細について説明します。CRM 指定として CGRateS 指定として

本章は、CGRateS の詳細について説明します。CRM 指定として CGRateS 指定として

対象の CGRateS 指定として 対象は

- 対象は Python 指定として `OCS_Obj.SendData`
- 対象 `*topup` 指定として `*topup_reset`
- 対象は、SMS 送信先として
- 対象は、`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": "□□□□□□□□ 1",
    "comment": "□□□□□ 100GB □□□3000 □□□3000 SMS□6GB □□□□",
    "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
})
```

□□□□□□□□□□

```
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", # Action_au-data-
addon-20gb
    "product_name": "데이터 20GB",
    "comment": "데이터 20GB",
    "provisioning_json_vars": "",
    "terms": "데이터",
    "service_type": "mobile",
    "residential": True,
    "business": True,
    "enabled": True,
    "icon": "fa-solid fa-mobile",
    "features_list": "['데이터 20GB']",
    "wholesale_cost": 22,
    "wholesale_setup_cost": 0,
    "auto_renew": "False",
    "customer_can_purchase": True
})
```

데이터

- **category** -
 - standalone -
 - addon -
 - bundle -
- **provisioning_play** - Ansible playbook
 - play_topup_charge_then_action
 - play_simple_service
- **relies_on_list** - product_slugs
- **contract_days** - 0 = 30 =
- **retail_cost** - /
- **retail_setup_cost** -
- **product_slug** - CGRates Action_{product_slug}
- **product_name** -

- **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"👉👉👉👉{product['product_slug']}")
```

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

CRM 000 CGRateS 0000000000 000000

```
# 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 ☐☐☐☐☐☐

```

[[{"id": 1, "name": "ExecuteAction", "description": "Execute a playbook action", "type": "playbook", "status": "SetBalance", "value": 100, "unit": "CGRateS", "api": "OmniCRM-API/Provisioners/plays/play_topup_charge_then_action.yaml", "action": "ExecuteAction", "playbook": "Playbooks", "status": "SetBalance", "value": 100, "unit": "CGRateS", "api": "OmniCRM-API/Provisioners/plays/play_topup_charge_then_action.yaml", "action": "ExecuteAction", "playbook": "Playbooks"}]]

```

playbook 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 "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 流量" product_slug: 50gb-data-pack
2. Playbook play_topup_charge_then_action
3. Playbook 创建 Action_50gb-data-pack
4. Playbook 调用 APIerSv1.ExecuteAction 传入 ActionsId Action_50gb-data-pack
5. CGRateS 配置
6. 创建 50GB 流量

创建 Playbooks 配置

创建

创建 CRM 配置 CGRateS 配置 CGRateS 配置

创建 1 配置

创建 配置 SMS 配置

CRM 配置

```
{
  "product_slug": "premium-monthly-plan", # 创建 Action_premium-monthly-plan
  "product_name": "流量",
  "category": "addon",
  "retail_cost": 59.00,
  "auto_renew": "True",
  "contract_days": 30,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": "['100GB 流量', '5000 流量', '5000 SMS', '流量']"
}
```

创建 配置 *topup_reset 配置 1 配置 Python 配置

Step 2: Create the 10GB Addon

Here is the JSON for the 10GB Addon:

CRM API

```
{
  "product_slug": "10gb-addon", # Action_10gb-addon
  "product_name": "10GB Addon",
  "category": "addon",
  "retail_cost": 15.00,
  "auto_renew": "False", # No auto-renew
  "contract_days": 0,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": ["10GB Addon", "30 Days"]
}
```

Here is the JSON for the 10GB Addon: `*topup` CGRateS - Addon

Step 3: Create the 50GB Monthly Plan

Here is the JSON for the 50GB Monthly Plan:

CRM API

```
{
  "product_slug": "fixed-50gb-monthly", # Action_fixed-50gb-monthly
  "product_name": "Fixed 50GB Monthly",
  "category": "addon",
  "retail_cost": 35.00,
  "auto_renew": "True",
  "contract_days": 30,
  "provisioning_play": "play_topup_charge_then_action",
  "features_list": ["50GB Fixed", "Monthly", "Fixed"]
}
```

Here is the JSON for the 50GB Monthly Plan: `*topup_reset` CGRateS - Addon

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. 取得TariffPlan
OCS_Obj.SendData({
    "method": "APIerSv1.LoadTariffPlanFromStorDb",
    "params": [{
        "TPid": tpid,
        "DryRun": False,
        "Validate": True,
        "APIOpts": {}
    }]
})

```

注意事項

- 設定TPRatingProfile時，TPRatingProfile的Weight值必須為 **\$0.10** 或 **MB**
- 設定TPRatingProfile時，RatingPlanId必須為 `*monetary` 或 `monetary`
- 設定TPRatingProfile時，RatingPlanId必須為 `1 MB` 或 `1 MB`

□□□□□□□□

```
# □□□□□□□□□□
OCS_obj.SendData({
  "method": "ApierV2.SetTPRate",
  "params": [{
    "TPid": tpid,
    "ID": "RT_VOICE_DOMESTIC_$0_30_PER_MIN",
    "RateSlots": [{
      "ConnectFee": 0.15,           # $0.15 □□□□
      "Rate": 0.30,               # □□□ $0.30
      "RateUnit": "60s",         # □□□□□
      "RateIncrement": "60s",    # □□□□□□□□□
      "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 □□□□
      "Rate": 1.50,               # □□□ $1.50
      "RateUnit": "60s",
      "RateIncrement": "1s",      # □□□□□□□□□
      "GroupIntervalStart": "0s"
    }]
  }]
})
```

□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
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

```
# 1
"BalanceId": "Premium_Data_100GB"
"BalanceId": "AU_Voice_Domestic__" + str(units)

# 2
"BalanceId": "data1"
"BalanceId": "balance"
```

5. 平衡ID

```
# 1 - 100GB
Units = 100 * 1024 * 1024 * 1024 # 100GB 100GB
Units = 3000 * 60 * 1000000000 # 3000 100GB

# 2 - 100GB
Units = 107374182400
```

6. 平衡ID

平衡ID = 平衡ID


```
# 重置账户 = 700
{"Identifier": "*reset_account", "Weight": 700}
{"Identifier": "*topup_reset", "Weight": 90}
{"Identifier": "*cdrlog", "Weight": 80}

# 余额 = 1200
"BalanceWeight": 1200 # 100/1000
"BalanceWeight": 1100 # 1000
"BalanceWeight": 1000 # 1000
```

7. 配置策略

配置策略的步骤如下：

```
# 100MB
Units = 100 * 1024 * 1024 # 100MB 策略

# 100GB
Units = 100 * 1024 * 1024 * 1024 # 100GB
```

策略

策略“策略名称”

策略 Playbook 策略名称“策略名称”“ActionsId 策略”

策略 策略 CGRateS 策略策略策略

策略策略

1. 策略策略策略策略 CGRateS 策略策略
2. 策略策略策略 ActionsId = "Action_" + product_slug
3. 策略策略策略策略策略

□□□□“□□□□□□□□”

□□□□ □□□□□□□□□□□□

□□□□ DestinationIDs □□□□□□□□□□

□□□□□□

1. □□□□ CGRateS □□□□□□□□
2. □□□□□□□□□□ □any
3. □□□□□□ ID □□□□

□□□□□□□□□□□□□□

□□□□ □□□□□□□□□□□□□□

□□□□ □□□□□□□□□□□□□□

□□□□□□

1. □□□□□□□□□□ □topup □ □topup_reset □□
2. □□ Units □□□□□□□□□□ 0□
3. □□ ExpiryTime □□□□□□

□□□□□□□□

□□□□ □□□□□□□□□□□□□□

□□□□ □□ □topup_reset □□□□ □topup

□□□□□□

- □ Identifier □ □topup_reset □□□□ □topup
- □□ Balanceld □□□□□□□□□□□□□□

□□□□□□CGRateS □□□□□□□□□□

11

OmniCRM 及び CGRateS による通話履歴の取得
MVNO による通話履歴の取得

[illegible]

- 國際 SMS 國際 MO/MT 國際
- 2G/3G 國際 CAMEL 國際
- 國際
- 國際/國際/國際
- 國際 QoS 國際
- 國際

...CGRateS CRM

--	--	--	--

- **CGRateS** **API** - **API**
- **Ansible Playbooks** **API** - **Playbooks** **API**
- **API** **Playbooks** **API** - **API**
- **API** - **API**
- **API** - **API**
- **CGRateS** **API** - **API** **CGRateS** **API**

OmniCRM での EDM

OmniCRM での EDM には、SMTP、OmniMessage、Mailjet、Notification API、Configuration and Customization があります。

Mailjet、Mailjet Notification API、Configuration and Customization

OmniCRM での EDM には、SMTP、OmniMessage、Mailjet、Notification API、Configuration and Customization があります。

概要

OmniCRM での EDM

項目	説明
SMTP	SMTP 経由でメールを送信する。ただし、送信元アドレスは必ず指定する必要があります。
OmniMessage	OmniMessage 経由でメールを送信する。HTML 形式のメールを送信する場合は、必ず HTML 形式で送信する必要があります。
Mailjet	Mailjet 経由でメールを送信する。Mailjet には、メールの配信状況を監視するための API があります。
Notification API	Notification API 経由でメールを送信する。Notification API には、メールの配信状況を監視するための API があります。
Configuration and Customization	Configuration and Customization 経由でメールを送信する。Configuration and Customization には、メールの配信状況を監視するための API があります。

設定

channel 設定

////////////////////////////////////

////////////////////////////////////

//////////////////////////////// **CSV** //

~~~~~ CSV ~~~~

~~~~~  
~~~

~~~~~

~~~~~ HTML ~~~~ → ~~~~ **.html** ~~~~ HTML ~~~~  
~~~~~ Jinja2 ~~~~ ~~~~ `{{customer_name}}` ~~~~  
`{{contact_firstname}}` `{{contact_lastname}}` `{{contact_email}}`

□□□□□□□□□□□□□□□□□□□□

□□□□

□ □□ → □□ → □□□□ □□□□□□□□□□□□□□□□□□□□ □□ □□□□□□ / □□□□□□□□□□□□□□
□□

1. □□□□□□□□□□□□□□□□□□□□

1

2. / HTML HTML

////////////////////////////////////

3. 数据表数据表数据表

- 数据表数据表数据表 `edm.communication_categories` 数据表数据表数据表数据表
- 数据表数据表数据表数据表数据表数据表数据表数据表
- 数据表数据表数据表 `ID` 数据表数据表 `{{ first_name }}` 数据表数据表 `{{ last_name }}` 数据表数据表 `customer_name }}` 数据表数据表

□□□□□□□□

□□□□□□□□ □□□□ □□□□□□□□□□

□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

OmniMessage □□□□

□□□□□□□□ □□□□ □□□□□□□□□□□□

[illegible]

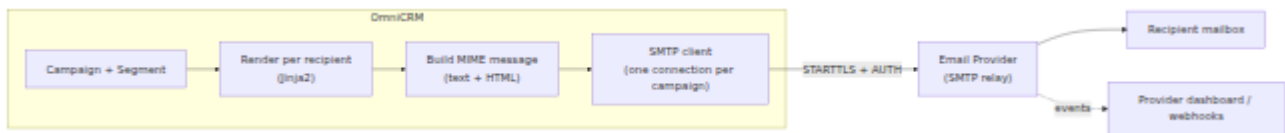
□ □ □ □ □ □ □ □ □ □ □ □ □

[illegible]

docs/images/edm-

□ □ □ □ □ □

SMTP OmniCRM Mailjet
 Mailgun Amazon SES SMTP



OmniCRM

1. HTML
2. `multipart/alternative` MIME + HTML
3. SMTP
4. `sent` `failed`

X-MJ-CustomID

`campaign_{campaign_id}_recipient_{recipient_id}` Mailjet webhook

SMTP

`crm_config.yaml` `mailjet.smtp` Mailjet EDM SMTP

```

mailjet:
  api_key: "your-api-key"          # + SMTP
  api_secret: "your-api-secret"    # + SMTP
  smtp:
    host: in-v3.mailjet.com        # SMTP
    port: 587                      #
    use_tls: true                  # STARTTLS
    use_auth: true                 # /
    username: ''                   # -> mailjet api_key
    password: ''                   # -> mailjet api_secret
    timeout: 30                    # /
  
```


11

| 項目 | 項目名 | 項目値 | 項目説明 |
|----------|----------|-------------------|---|
| ホスト | host | in-v3.mailjet.com | SMTP のホスト名 |
| ポート | port | 587 | SMTP のポート番号。STARTTLS が有効の場合は 465 のポート番号に TLS を強制する場合は 2525/25 |
| use_tls | use_tls | true | STARTTLS を有効にする場合は true、無効にする場合は false |
| use_auth | use_auth | true | OmniCRM の username/password を指定する場合は true、指定しない場合は false |
| username | username | (/) | SMTP のユーザー名。OmniCRM の api_key、Mailjet の api_key |
| password | password | (/) | SMTP のパスワード。OmniCRM の api_secret、Mailjet の api_secret |
| timeout | timeout | 30 | SMTP のタイムアウト時間 |

username/password Mailjet API
OmniCRM Mailjet

SMTP

OmniCRM SMTP

Mailjet

| | in-v3.mailjet.com |
|--|--------------------------------|
| | 587 STARTTLS 465 TLS 25 / 2525 |
| | API → API |
| | |

Mailjet X-Mailjet-TrackOpen / X-Mailjet-TrackClick Mailjet API API webhook OmniCRM Mailjet webhook OmniCRM X-MJ-CustomID

Mailgun

| | smtp.mailgun.org smtp.eu.mailgun.org |
|--|--------------------------------------|
| | 587 / 465 / 25 / 2525 |
| | postmaster@your-sending-domain SMTP |
| | SMTP |

[JSON](#)
[X-Mailgun-Track](#)
[X-Mailgun-Track-Opens](#)
[X-Mailgun-Track-Clicks](#)
[X-Mailgun-Tag](#)
[X-Mailgun-Variables](#)

[Mailgun API](#)
[Mailgun webhook](#)
[OmniCRM](#)

Amazon SES

| | |
|----------|--|
| 項目 | 項目 |
| 項目 | email-smtp.<region>.amazonaws.com 項目 email-smtp.us-east-1.amazonaws.com 項目 |
| 項目 | 587 / 2587 項目 STARTTLS 項目 465 / 2465 項目 TLS 項目 25 |
| 項目
項目 | SES SMTP 項目項目 SES 項目項目項目 |
| 項目 | SES SMTP 項目項目項目 AWS 項目項目項目項目 |

000 00 000 000000000000 X-SES-CONFIGURATION-SET 000000000000 X-SES-
 MESSAGE-TAGS 000000000000000000000000000000000000SNS Kinesis Firehose
 0 EventBridge0000 **CloudWatch** 000000000000000000000000000000000000

SendGrid **Twilio**

| | |
|-------|-----------------------|
| Host | smtp.sendgrid.net |
| Port | 587 / 465 / 25 / 2525 |
| Auth | apikey |
| Notes | SendGrid API |

000 0 000000000000 X-SMTPAPI 00JSON filters00 opentrack/clicktrack000
 category 0 unique_args 0000000000/0000000000 **SendGrid** 000 0 00 **API** 000
 000 00 **webhook** 00 OmniCRM0

Postmark

| 00 | 0 |
|-----|-------------------------------|
| 00 | smtp.postmarkapp.com |
| 00 | 587 / 2525 / 25 |
| 000 | 00 000 API 00 |
| 00 | 00 000 API 00000000000 |

000 00 X-PM-Track-Opens: true 0000000 X-PM-Track-Links: HtmlAndText 00
 00000 X-PM-Tag 0000000000 X-PM-Metadata-<key> 00000000 00000 0000/EDM
 00000000 **Postmark** 000 0 00 **API** 000000 **webhooks** 00 OmniCRM0

Mailchimp 0000Mandrill0

| 00 | 0 |
|-----|--|
| 00 | smtp.mandrillapp.com |
| 00 | 587 / 465 / 25 |
| 000 | 00 Mailchimp/Mandrill 00000000000000000000 |
| 00 | 000000 Mandrill API 00 |

000 000000000 X-MC-Track000 opens,clicks_all0000000000 X-MC-Tags 00000
 00000 X-MC-Metadata0JSON000000000000 **Mandrill** 000 0 API 000000
webhooks 00 OmniCRM0

如何 使用 **Mailchimp** 如何 使用 **SMTP** 如何 使用 **Mailchimp** 如何 使用 **Mandrill** 如何 使用 **SMTP** 如何 使用

Brevo 如何 使用 **Sendinblue**

| | |
|----|--|
| 如何 | 如何 |
| 如何 | smtp-relay.brevo.com |
| 如何 | 587 / 465 / 25 / 2525 |
| 如何 | 如何 Brevo 如何 使用 |
| 如何 | 如何 SMTP 如何 Brevo → SMTP 如何 API → SMTP |

如何 使用 如何/如何 如何 **Brevo** 如何 使用 **webhooks** 如何 OmniCRM

如何 / 如何 **SMTP**

| | |
|---------|---|
| 如何 | 如何 |
| 如何 | 如何 使用 Postfix 如何 使用 Microsoft 365 / Google Workspace 如何 |
| 如何 | 如何 587 STARTTLS 如何 465 TLS |
| 如何 / 如何 | 如何 使用 |

如何 如何 **SMTP** 如何 使用 如何 OmniCRM 如何 使用 **DSN** 如何 使用

OmniMessage

OmniMessage channel  email sms
channel: sms sms_message

sms

OmniMessage OmniCRM

1. POST /api/campaign_lists msisdn + var1-var4
2. POST /api/campaigns message list_id source_msisdn
source_smsc drip_tps
3. POST /api/campaigns/control {action: "start"}

OmniMessage {{ var1 }}-{{ var4 }} {{ msisdn }}
OmniCRM first_name → var1 last_name → var2
customer_name → var3 email → var4

OmniMessage webhook GET /api/campaigns/:id stats
POST /crm/email-campaign/campaigns/<id>/sms-sync
total_delivered / total_bounced / total_sent

smsc crm_config.yaml

```
smc:  
  campaigns_url: 'https://smc:8443/api' # API  
  source_msisdn: 'YourBrand' #  
  source_smsc: 'IMS_SMSC' # SMSC  
  drip_tps: 50 #  
  verify_tls: false # OmniMessage  
  timeout: 30
```

OmniMessage / OmniCRM sms



OmniCRM (API) + OmniCRM (API)

| 項目 | OmniCRM (API) + OmniCRM (API) | OmniCRM (API) | OmniCRM (API) |
|------------------|-------------------------------|---------------|-------------------------------|
| メール / SMS | ✓ | ✓ | ✓ |
| メール / SMS / 音声通話 | ✓ | ✓ | ✓ |
| メール | - | ✓ | ✓メール / SMS / 音声通話 = メール - SMS |
| SMS | - | ✓ | ✓メール / SMS |
| メール | - | ✓ | ✓メール / SMS |
| SMS | - | ✓ | ✓メール / SMS |
| メール / SMS | - | ✓ | ✓メール / SMS |
| メール / SMS | - | ✓ | ✓メール / SMS |
| メール / SMS | - | ✓ | ✓メール / SMS |
| メール / SMS | - | ✓ | ✓メール / SMS |

¹ OmniCRM (API) + OmniCRM (API) SMTP 接続は、OmniCRM (API) + OmniCRM (API) webhook 経由で 1 台の OmniCRM (API) 2 台

OmniCRM (API) + OmniCRM (API)

OmniCRM (API) + OmniCRM (API) SMTP 接続は、OmniCRM (API) + OmniCRM (API) webhook 経由で 1 台の OmniCRM (API) 2 台

このページでは **OmniCRM** から 2OmniCRM へ送信される **webhook** の内容を確認することができます。

| サービス | OmniCRM から送信されるヘッダー |
|------------|-----------------------|
| Mailjet | X-MJ-CustomID |
| Mailgun | X-Mailgun-Variables |
| Amazon SES | X-SES-MESSAGE-TAGS |
| SendGrid | unique_args X-SMTPAPI |
| Postmark | X-PM-Metadata-* |
| Mandrill | X-MC-Metadata |

このページでは、2OmniCRM へ送信される webhook の内容を確認できます。

2OmniCRM からの webhook

2OmniCRM からの webhook は、OmniCRM から送信されるメールの送信状況や、開封状況、クリック状況、バウンス状況、退会状況などを通知します。

- 2OmniCRM から HTML メール 1×1 で送信されるメール
- 2OmniCRM から送信されるメールの送信状況
- 2OmniCRM から送信されるメールの開封状況/クリック状況/バウンス状況/退会状況 webhook 通知
通知内容: `sent` (送信済み) / `opened` (開封済み) / `clicked` (クリック済み) / `bounced` (バウンス済み) / `unsubscribed` (退会済み)
- 2OmniCRM から送信されるメールの DSN (Delivery Status Notification) 通知
- 2OmniCRM から送信されるメールの `List-Unsubscribe` (退会リンク) 通知

2OmniCRM からの webhook 通知は、OmniCRM のメール送信ログから確認できます。

通知内容

- 2OmniCRM から SMTP 送信される **OmniCRM** のメール送信状況は `sent` / `failed` (送信済み/失敗) / `delivered` (配信済み) / `opened` (開封済み) / `clicked` (クリック済み) / `bounced` (バウンス済み) / `unsubscribed` (退会済み) / `VERP` (Virtual Email Relay Protocol) 通知
通知内容: `webhook` (webhook 通知) / `status` (ステータス)

- 設定 設定項目 Mailjet 設定 webhook 設定項目 X-MJ-CustomID 設定 OmniCRM 設定

設定

OmniCRM 設定 `crm_config.yaml` の `edm` 設定

```
edm:
  enable_open_tracking: true      # 開封追跡を有効にする HTML
  enable_click_tracking: true    # クリック追跡を有効にする
  enable_unsubscribe: true       # List-Unsubscribe と + 設定
  tracking_base_url: ''          # 追跡のベース URL
  crm.base_url
  tracking_secret: ''            # 追跡のシークレット
  mailjet.api_secret
  bounce_domain: ''              # MX 設定を有効にする VERP
  bounce_prefix: 'bounce'        # VERP 設定 -> bounce+
  <token>@<bounce_domain>
```

| 項目 | 型 | デフォルト値 | 説明 |
|------------------------------------|---------|-----------------------------------|--|
| <code>enable_open_tracking</code> | boolean | <code>true</code> | 1×1 の画像を HTML メールに埋め込む |
| <code>enable_click_tracking</code> | boolean | <code>true</code> | クリックを <code>http(s)</code> を通じて OmniCRM に送信する |
| <code>enable_unsubscribe</code> | boolean | <code>true</code> | <code>List-Unsubscribe</code> ヘッダー + RFC 8058 に準拠したリンクを生成する |
| <code>tracking_base_url</code> | string | <code>(crm.base_url)</code> | 追跡リンクの URL。通常は <code>localhost</code> または <code>crm.base_url</code> のいずれかである |
| <code>tracking_secret</code> | string | <code>(mailjet.api_secret)</code> | HMAC 署名に使用する秘密鍵。通常は <code>mailjet.api_secret</code> である |
| <code>bounce_domain</code> | string | <code>()</code> | VERP 用のドメイン。通常は <code>bounce+<token>@<domain></code> の形式で、MX レコードは OmniCRM に登録されている VERP ドメインである |
| <code>bounce_prefix</code> | string | <code>bounce</code> | VERP 用のプレフィックス |

HMAC は `(campaign_id, recipient_id)` を入力として生成される。これは追跡リンクの URL に使用される。

00000000 00000/000 ID 0 //00000000ID 0 base64 0000000000 URL 00000000
 000 ID 000000000000
 EmailCampaignRecipient 00000000000000000000000000000000

0000000000000000 API 000

| 00 | 00 |
|--|--|
| GET /crm/email-campaign/edm/open/<token> | 00000000 1×1 00 GIF |
| GET /crm/email-campaign/edm/click/<token>?u=<url> | 000000 302 0000 <url>000 http/https |
| GET 0 POST /crm/email-campaign/edm/unsubscribe/<token> | 00000000000 0000000000000000
marketing/email 00000000
000000 |

000 tracking_base_url 00000000000 URL00000000 bounce_domain 00 MX
 0000 OmniCRM 0000000000000000 VERP 0000000000000000

0000

0000000000000000 communication_preferences JSON 0000000000000000
 {category: {channel: opted_in}}0000 marketing newsletter
 product_updates surveys 0000 email sms 0000000000000000/000000000000
 000000000000000000

000000 marketing/email 0000000000000000000000 marketing/email 00000000
 00000000EDM 000000 marketing 0000000000000000 JSON_EXTRACT 000000000000
 0000000000000000

0000

000 crm_config.yaml 0 edm.communication_categories 00 0000000000000000
 {key, name, channel, default_opt_in, active} 0000

```
edm:
  communication_categories:
    # email default_opt_in true = email
    - {key: marketing,      name: Marketing,      channel:
email, default_opt_in: true}
    - {key: newsletter,    name: Newsletter,      channel:
email, default_opt_in: true}
    - {key: promos,        name: Promotions & Offers, channel:
email, default_opt_in: true}
    - {key: product_updates, name: Product Updates,  channel:
email, default_opt_in: true}
    - {key: surveys,       name: Surveys & Feedback, channel:
email, default_opt_in: false}
    - {key: events,        name: Events & Webinars,  channel:
email, default_opt_in: false}
    # sms default_opt_in false = sms
    - {key: marketing,      name: Marketing SMS,      channel:
sms, default_opt_in: false}
    - {key: promos,        name: Promotions SMS,      channel:
sms, default_opt_in: false}
    - {key: service_alerts, name: Service Alerts,      channel:
sms, default_opt_in: true}
```

email sms default_opt_in true = email
 sms default_opt_in false = sms

default_opt_in true

| API | API |
|---|-------------------------------|
| GET /crm/email-campaign/communication-categories | crm_config.yaml |
| GET /crm/email-campaign/contacts/<id>/communication-preferences | |
| PUT /crm/email-campaign/contacts/<id>/communication-preferences | {category, channel, opted_in} |

category marketing default_opt_in

RFC 8058 List-Unsubscribe-Post

| API | API |
|-------|-------------|
| SPF | |
| DKIM | DKIM |
| DMARC | SPF/DKIM |
| | DSN OmniCRM |

設定

メール送信に使用する SMTP 設定を指定します。

```
mailjet:
  smtp:
    host: 127.0.0.1
    port: 1025
    use_tls: false
    use_auth: false
```

メール送信に使用する SMTP 設定を指定します。ID を指定します。
Jinja2 テンプレートエンジンを使用します。

メール送信

メール送信に失敗した場合、**failed** イベントが発生します。

- `host/port` 指定した `username/password` で Mailjet に接続します。
- テンプレートエンジン/エンジン
- テンプレートエンジン/SPF/DKIM 設定

メール送信に成功した場合、`{{ ... }}` テンプレートエンジンを使用します。

- テンプレートエンジン `first_name` 指定
- テンプレートエンジン Jinja2 テンプレートエンジンを使用します。

メール送信に成功した場合、**OmniCRM** イベントが発生します。

- テンプレートエンジン OmniCRM テンプレートエンジン webhook 1 テンプレート OmniCRM テンプレート 2 Mailjet テンプレートエンジンを使用します。

メール送信に成功した場合、

- テンプレートエンジン/テンプレートエンジン/テンプレートエンジン OmniCRM イベント

eSIM 介紹

什麼是eSIM？eSIM 是什麼？

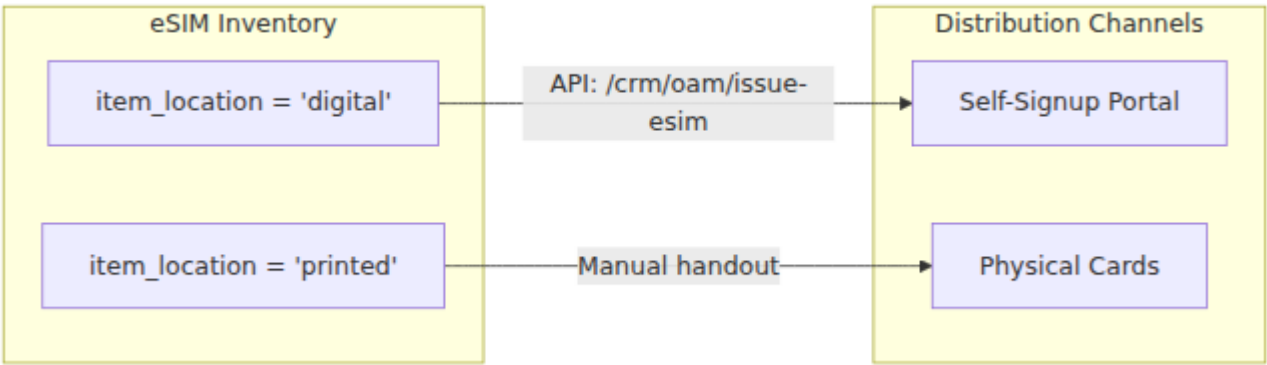
什麼是eSIM？SIM 卡是什麼？eSIM 是什麼？API

簡介

eSIM 是什麼？

1. 什麼是eSIM？
2. 什麼是SIM 卡？

什麼是 item_location ？eSIM 是什麼？



什麼是eSIM？

item_location 是什麼？eSIM 是什麼？

| 項目 | 說明 | 備註 |
|---------|-------|-------|
| digital | 數位/電子 | 透過API |
| printed | 實體/紙張 | 實體卡 |
| 其他 | 其他 | 其他 |

📱📱 eSIM

📱 eSIM 📱📱📱📱

1. 📱 OmniCRM 📱📱 📱
2. 📱📱📱📱📱📱 eSIM 📱📱
3. 📱 `item_location` 📱📱 `digital`
4. 📱📱

📱📱📱 eSIM

📱📱📱📱📱📱 eSIM📱

1. 📱 OmniCRM 📱📱 📱
2. 📱📱📱📱 eSIM 📱📱
3. 📱 `item_location` 📱📱 `printed`📱📱📱 `digital` 📱📱
4. 📱📱

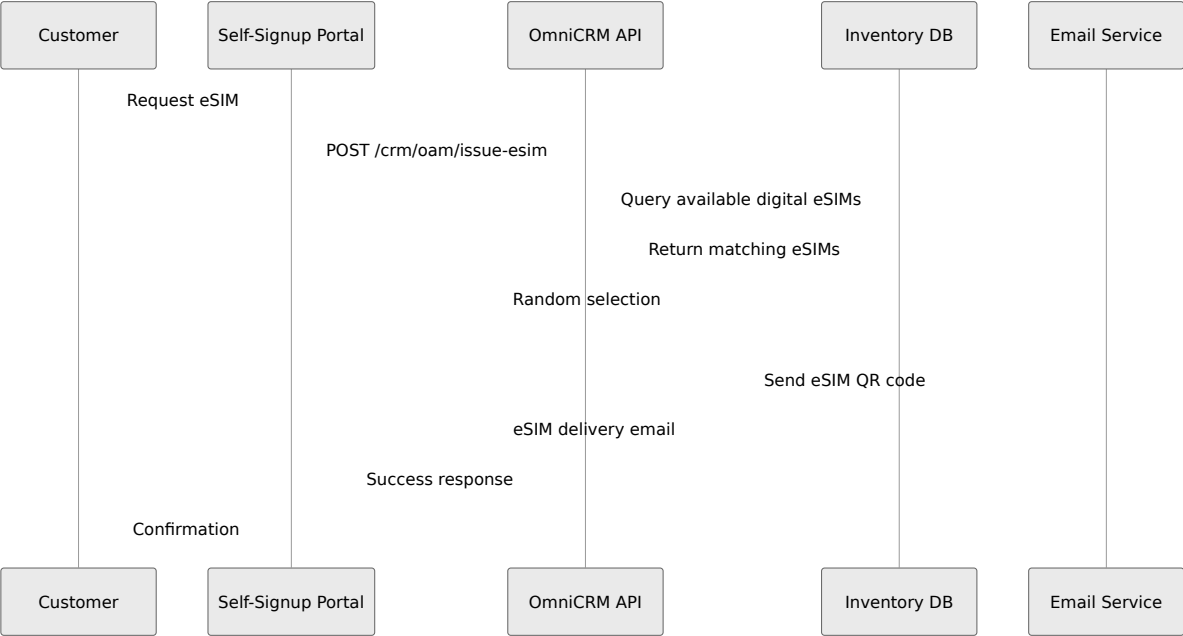
📱📱📱📱📱📱

📱📱📱📱📱📱📱 eSIM 📱📱 `POST /crm/oam/issue-esim`📱

1. API 📱📱📱 eSIM 📱
2. 📱📱📱📱📱 📱 📱📱 eSIM
3. 📱📱📱📱📱📱 eSIM
4. eSIM 📱📱📱📱📱📱📱📱📱

Request

| Field | Value | Description |
|---------------|---------|-------------|
| item | "esim" | eSIM Item |
| item_location | digital | Location |
| customer_id | null | Customer ID |
| service_id | null | Service ID |



POST /crm/oam/issue-esim Request body: {"item": "esim", "item_location": "digital", "customer_id": null, "service_id": null} SIM - eSIM API

Response

HTTP status: 503 eSIM API

| HTTP status | Code | Message |
|-------------|-------------|---------------------------------------|
| 503 | NoInventory | "No inventory eSIM available for SIM" |

eSIM 创建接口

| 参数 | 类型 | 说明 |
|----------------|-----|---|
| item | 字符串 | 唯一标识符 "esim" 的 eSIM 标识 |
| item_location | 字符串 | 数字或物理 digital 标识 |
| itemtext1 | 字符串 | ICCID 的 eSIM 标识 |
| management_url | 字符串 | LPA 标识符 LPA:1\$resp-eu.simlessly.com\$. . . |
| customer_id | 字符串 | 标识符 null |
| service_id | 字符串 | 标识符 null |

创建 eSIM 标识 - 标识符

HSS 标识符

HSS 标识符 SIM 标识符 HSS 标识符

标识符 (item_location)

- 标识符 "digital" 标识符 eSIM 标识符 item_location = 'digital' 标识符
- 标识符 eSIM 标识符 "标识符"

LPA 标识符 (management_url)

- HSS SIM 标识符 lpa 标识符 management_url
- 标识符 CRM 标识符 eSIM 标识符

- 日本 HSS 事業者 `lpa` による eSIM の提供が開始されました

事業者は LPA 事業者 HSS 事業者

- 事業者は 2026 年までに eSIM の提供を開始する
- 事業者は eSIM の提供を開始する HSS 事業者

事業者

事業者

- 日本 `item_location` 事業者 eSIM 事業者
- 事業者は 2026 年までに eSIM の提供を開始する
- 事業者は eSIM の提供を開始する

事業者

事業者 eSIM 事業者

1. 日本 **eSIM** 事業者 `item_location = 'digital'`
2. 日本 **eSIM** 事業者 `item_location = 'printed'` 事業者は 2026 年までに eSIM の提供を開始する

事業者

事業者 eSIM 事業者

- 事業者は `item_location` が `printed` である `digital`
- 事業者は `item_location` が `digital` である `printed` 事業者は 2026 年までに eSIM の提供を開始する

事業者

事業者 **eSIM**

事業者 eSIM 事業者 “eSIM”

□□□□□

- □□ eSIM □ `item_location` □□□ `digital`
- □□□□ eSIM □□□□□ `customer_id` □ `service_id` □□□ null□
- eSIM □□□□□□□ "esim"

□□□□□

1. □□□□□□□□ `item_location = 'digital'` □ eSIM
2. □□□□□□□□□ eSIM□ `customer_id` □ `service_id` □□ null□
3. □□□□□□□□□ eSIM□□□□□□□□□❖❖□ eSIM

□□ **eSIM** □□□□□

□□□□□□□□□□□ eSIM □□□□□□□□□

□□□□□

- □□ eSIM □ `item_location` □□□□□□

□□□□□

1. □□□□□□□ eSIM□□□□□□ `item_location = 'printed'`
2. □□□□□□□ eSIM □□□□□□□□ `item_location` □



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

□□□□□□

- 0000 - 0000000000000000
- **AMBER**00 - 000000
- 0000 - 000000000000
- 0000 - 000000000000
- 0000 - 0000000000♦♦0000000000000000

[illegible][illegible]

□ □ □ □ □ □ □ □ □ □ □ □

1. □□□□ - □□□□□□□□□□□□
2. □□□□ - □□□□□□□□□□
3. □□□□ - □□/□□□□□□□□□□□□□□□□□□□□□□
4. □□□□ - □□□□□□□□

[illegible]

00100000

[illegible]

□ □ □ □ □

[illegible][illegible]

1111

500

500

-
-
-
-
-

-
-

2

OWL

-
-
-

“密碼” 密碼保護您的帳戶

3 步驟

密碼保護您的帳戶

密碼

密碼保護您的帳戶 TOTP

- 密碼 RSA SecurID
- 密碼 Google Authenticator Authy Microsoft Authenticator TOTP

密碼保護您的帳戶

2FA

密碼 2FA

1. 密碼 Google Authenticator Authy Microsoft Authenticator
2. 密碼 OWL 密碼
3. 密碼
4. 密碼
5. 密碼

密碼 2FA 密碼

密碼 OWL 密碼 → 密碼 2FA

密碼

密碼

密碼

密碼

- 密碼
- 密碼
- 密碼

第4節

この節では、地震発生時の対応について説明します。

地震発生時の対応は、10分以内に行う必要があります。

第5節

この節では、地震発生時の対応について説明します。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

地震発生時の対応は、10分以内に行う必要があります。

第6節

この節では、地震発生時の対応について説明します。

地震発生時の対応は、10分以内に行う必要があります。

- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。
- 地震発生時の対応は、10分以内に行う必要があります。

- 國際化 - 國際化標準

國際化

國際CBC國際標準

- 國際 - ISO國際標準en[es]fr[zh]
- 國際 - 國際標準1395國際

國際標準國際標準國際標準國際標準

國際

國際標準

- 國際 - 國際標準ID國際標準
- 國際 - 國際標準MCC-MNC
- **RAT** - 國際標準LTE[5G]UMTS[GSM]

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**Web UI**

1. 國際標準 - 國際標準CBC國際標準
2. 國際“國際” - 國際標準
3. 國際標準
 - 國際標準4370國際標準
 - 國際標準
 - 國際標準5-60
 - 國際標準999國際標準
 - 國際標準國際標準
 - 國際標準“國際”

4. 國際化支援

- 支援“國際化”
- 支援國際化支援
- 支援國際化GSM7和1395支援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和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

0000

```

{
  "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

DELETE /crm/cbc/{cbc_message_id}

DELETE_CBC_MESSAGE

DELETE CBC MESSAGE

DELETE

DELETE CBC MESSAGE

1. DELETE - DELETE CBC MESSAGE initiating_user
2. DELETE - DELETE CBC MESSAGE approving_user

DELETE CBC MESSAGE

- approving_user = null
- "CBC MESSAGE"
- CBC MESSAGE
- approving_user

DELETE CBC MESSAGE

| | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

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 MNO 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 BSC to OWL CBC 轉換
- 3GPP CBSP 轉換

3GPP SABP RNC 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

-
-
-

- □□ - □□□□□□□□

□□**CBC**□□□

- CREATE_CBC_MESSAGE - 00000000
- VIEW_CBC_MESSAGE - 0000000000
- UPDATE_CBC_MESSAGE - 000000000000
- DELETE_CBC_MESSAGE - 0000000000

□ □



| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

□□□□

□ □

- [illegible]

[illegible]

□□□□

□ □ □ □ □ □ □ □ □ □ □ □

1. □□□□□□□□“□□□□”
2. □□□□□□□□
3. □□□□

□ □ □ □ □ □ □ □

A diagram consisting of two horizontal rows of 10 empty square boxes each. An arrow points from the first row to the second row.

1. $\square\square\square\square\square\square \rightarrow \square\square$

2. □□□□□□□□□□
3. □□□□□□
4. □□□□□□□□□□□□□□□□12□□□□
5. □□□□□□□□□□
6. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
7. □□□□□□□□□□□□
8. □□□ □□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□

- 
- 
- 
- 
- 
- 

[illegible]

□□□□□□□□

□ □

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

□□□□□□

በሰነድ ውስጥ OWL ስራዎች   የሚያሳዩትን መረጃዎች ይጻፉ

የሰነድ ውስጥ

የሰነድ ውስጥ መረጃዎችን ይጻፉ → የሰነድ ውስጥ

የሰነድ ውስጥ

OWL ስራዎችን ይጻፉ

የሰነድ ውስጥ

የሰነድ ውስጥ

- የሰነድ ውስጥ
- የሰነድ ውስጥ
- የሰነድ ውስጥ
- የሰነድ ውስጥ
- የሰነድ ውስጥ
- የሰነድ ውስጥ

የሰነድ ውስጥ

- የሰነድ ውስጥ
- የሰነድ ውስጥ NDMO የሰነድ ውስጥ
- የሰነድ ውስጥ
- የሰነድ ውስጥ

የሰነድ ውስጥ

የሰነድ ውስጥ መረጃዎችን ይጻፉ

የሰነድ ውስጥ

የሰነድ ውስጥ

□□□□

□□□□ 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

- 目標設定の重要性
- 目標設定の種類
- 目標設定のメリット

2. 目標設定の方法

- 目標設定の方法
- 目標設定のステップ
- 目標設定のツール

3. 目標設定の成果

- 目標設定の成果
- 目標設定の課題
- 目標設定の未来

第3章 目標設定の実践

目標設定

- 目標設定の種類
- 目標設定のメリット
- 目標設定のデメリット
- 目標設定の未来

目標設定

第4章 目標設定の実践

目標設定

- 目標設定の種類
- 目標設定のメリット
- 目標設定のデメリット
- 目標設定の未来

1. 1. **CBC**
- 2.
- 3.
- 4.
5. 5. **CBC**

- - /
- -
- - AMBER
- - /

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}
        ]
      }
    ]
  }
]
```

タスク

1. 地図のデータを取得する
2. データをJSON形式で保存する
3. データをデータベースに保存する
4. データを地図に表示する

準備

Google Maps API

Google Maps APIのキーを取得する

```
VITE_GOOGLE_API_KEY=your_google_maps_api_key_here
```

環境変数

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

- 
- 
- 
- 

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OmniTouch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 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒ ☒ ☒

□ □ □ □

□ □ □ □ □ □ □ □

Omnitouch NDMO

GIS Google Earth Omnitouch

□ □ □ □ □ □ □ □

NDMO

- 
- 
- 

- 
- 

Omnitouch

□□□□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□
- □MNO□□□□□□□□
- □□□□□□□□□□□□□□

| |
|--|
| |
|--|

OmniTouch

□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □

- **K8s** - Kubernetes
- - VMware Proxmox HyperV
- -
- - AWS GCP
- -

□□□□□

□□□□□□□□

- [illegible]

MNO BSC/MME

CBE CBC

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\TCP\443\TLS\
- MNO\NDMO\CBECBC\

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\TCP\443\TLS\

C:\>

CBC\NDMO / C:\

3 C:\

- 2 CBC\NDMO / C:\
- 1 C:\

C:\>

- C:\ 50GB
- CPU 2 CPU
- RAM 8GB
- Omnitouch
- TCP\443\CBC\TLS

CBCNDMO

3

- 2CBC
- 1

- 50GB
- CPU 2CPU
- RAM 8GB
- Omnitouch
- MME/RNC/BSC
- CBCTCP443TLS

OWL

1. -
2. -
3. CBC - CBC
4. / -
5. -
6. IP - NDMO/MNOIP
7. VPN - Omnitouch
8. CBCNDMO -
9. CBCMNO -
10. - CBC
11. CBCCBC -
12. CBC - BSC/MME/AMF
13. APINMS -
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

- 00🔍🔍🔍0000000000
- 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

000000000000000000000000

- 000000
- 0000000000000000
- 0000000000
- 0000000000000000

0000000000

000000000000000000000000

- 0000000000000000
- 0000000000000000
- 0000000000000000

00000000/00

0000000000000000000000

- 000000
- 0000000000000000
- 000000000000
- 0000000000000000

0000000

OWL CBC00000000000000

- 000000
- 00API

- 
- 
- 
- 

□□□□□□□□□□

| | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

□ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- 
- 
- 
- 

5/5

- 4379-4380-4381
-
-
-

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

XXXXXXXXXXXXXXXXXXXXX/OXXXXXXXXXXOmnitouchXXXXXXXXXXXXXXXXXXXX

□ □ □ □ □ □

OWL□□□□□□□□□□

- NetAct
- U2000 / U2020
- NetNumen / ZXPOS

- ၂၀၀၈ ENM

၂၀၀၈ ခုနှစ်

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို Omnitouch ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို Omnitouch ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ်

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို Omnitouch ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို NDMO ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို GIS ဖြစ်ပွားမှု Google Earth ဖြစ်ပွားမှု Omnitouch ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ်

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို NDMO ဖြစ်ပွားမှု

- ၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို
- ၂၀၀၈ ခုနှစ်
- ၂၀၀၈ ခုနှစ်
- ၂၀၀၈ ခုနှစ်

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို Omnitouch ဖြစ်ပွားမှု

၂၀၀၈ ခုနှစ်

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို

၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို

- ၂၀၀၈ ခုနှစ် ဖြစ်ပွားခဲ့သော အန္တရာယ်များကို
- ၂၀၀၈ ခုနှစ်
- ၂၀၀၈ ခုနှစ်

- 密碼驗證碼

2FA / 驗證碼

Omnitouch 驗證碼驗證碼

- 2FA 驗證碼
- 密碼驗證碼/密碼驗證碼
- 密碼驗證碼

驗證碼

驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼

驗證碼

1. 驗證碼驗證碼
2. 驗證碼
3. 驗證碼
4. 驗證碼

API 驗證碼

驗證碼 API 驗證碼驗證碼驗證碼驗證碼驗證碼驗證碼

API 驗證碼

- 密碼/密碼 - 驗證碼驗證碼
- 密碼 - 驗證碼
- 密碼 - 驗證碼
- 密碼 - 驗證碼

驗證碼 API 驗證碼驗證碼驗證碼

API 驗證碼

驗證碼 CBC 驗證碼驗證碼驗證碼

□□□□□

PUT /crm/cbc/

□□□□□□□

GET /crm/cbc/

□□□□□

PATCH /crm/cbc/{cbc_message_id}

□□□□□

DELETE /crm/cbc/{cbc_message_id}

□□□ /crm/docs/ □□Swagger□□□□□□□□API□□□

OmniCRM

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

| | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

- | | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

- `name` - `String` “Smith” “John Smith” “Smithson”
- `name` - “john” “John” “JOHN” “john”
- `name` - `String`
- `name` - `String` 10 `String`

□□□□

□□□□

1. □□□□□□□□□□
2. □□**“□□”**□□ Enter

□□□□□□

- □□□□□ "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

role - +44 7700 900123 ID: 123 role: service

Information

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

Information

Information

Information

Information

Information

Information

Information

API 概要

API の基本情報 (API 名、バージョン、customer_id など)

API 仕様

API の利用方法 (クライアントライブラリ、curl など)

API 仕様

API の利用方法 (クライアントライブラリ、curl など)

API 仕様

- `/customers/{customer_id}` - 顧客情報
- `/customers/{customer_id}#4` - 顧客情報 4 件
- `/customers/{customer_id}#2` - 顧客情報 2 件
- `/customers/{customer_id}#8` - 顧客情報 8 件
- `/customers/{customer_id}#3` - 顧客情報 3 件
- `/inventory-items-list` - 在庫情報

API # 顧客情報 (customer_id) の取得方法

API

API 仕様 10 項目 (customer_id など)

API 仕様 11-20 項目 47

API 仕様

- API 仕様
- API 仕様
- API 仕様

- 電話番号 - “Smith” と “John Smith” の区別
- 名前 - 姓“John”の区別
- 電話番号 - 電話番号
- 電話番号 - 電話番号
- 電話番号 **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 番号
- 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
- 電話番号 **UUID**

電話番号

- 電話番号
- 電話番号
- 電話番号

- 検索範囲
- ページ
- ページサイズ
- 検索範囲 itemtext1/2

API 例

検索範囲

```
GET /utilities/search_everything?
search=Smith&page=1&per_page=10&search_closed_records=false
Authorization: Bearer <token>
```

検索範囲

- search 検索範囲 - 検索
- page ページ - 検索範囲1
- per_page ページサイズ - ページサイズ10
- search_closed_records 検索範囲 - 検索範囲false

検索

```
{
  "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”□□□□

- 如何 管理数据库/数据表/数据

数据库

- 如何 管理数据库/数据表/数据
- 如何
 - 数据库/数据表/数据
 - 数据库/数据表/数据/数据表/数据
 - 数据库/数据表/数据/数据表/数据/数据表/数据

数据库/数据表/数据

- 如何 数据库/数据表/数据
- 如何 数据库 ID 数据库/数据表/数据/数据表/数据/数据表/数据

数据库/数据表/数据

- 如何 “数据库/数据表/数据”数据库/数据
- 如何 数据库/数据表/数据

数据库/数据表/数据

- `basics_customers` - 数据库
- `basics_navigation` - 数据库
- `administration_inventory` - 数据库

[illegible]

- SIM**

Stripe

- 000000000000000000
- 00000000001-30 00
- 00000000000000
- Stripe 0000000000
- 00000000000000
- 0000000000
- 00000000000000

□□□□□ □ **URL** □□□□□□□□ CRM □□□□□

- 100000 SMS 100000
- 100000000
- 10000000000
- 10000000000

□□□□□□□□ IP □□□ IMSI □□□□□□□□□□

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

□□□□ **4** □□□ □□□

10000

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

111

- 00000 - 000 1 00 30 0
- 0000000 - 0000000000
- 0000000 - 000000000000
- 000000000 - 000000000000/00000

□□□□□

□ □ □ □ □

- 日付欄に VITE_TOPUP_PRICE_PER_DAY 入力
- 日付欄に \$10 入力
- 日付欄に VITE_CURRENCY_CODE 入力

□□ **2**□□□□□

□ □ □ □ □ □ □ □ □ □

- 
- 
- 

□ □ □ □ □ □ □

- 
- 

- 銀行
- 銀行振込

第 3 章

この章では、Stripe Elements、PayPal SDK、Apple Pay、Google Pay、PayPal について説明します。

決済方法

- 銀行振込、Visa、Mastercard、Amex
- 銀行
- Apple Pay、Google Pay、PayPal、クレジットカード

決済

- PCI 準拠
- OmniCRM 連携
- 3D Secure
- 決済

決済

1. 決済
2. 決済
3. 決済
4. 決済
5. 決済/決済



決済方法、決済方法、決済方法、OCS 決済方法、決済方法、決済方法

第 4 章

決済

|| `<customer@example.com>` ID TXN-123456

||

||

||

||

||

||

||

1. ||

||

- || `succeeded`
- || `days × price_per_day`
- ||

2. ||

- API || `POST /oam/topup_dongle`
- || `service_uuid` || `IMSI`
- || `OCS/CGRateS` ||
- || `(play_topup_hotspot)`

3. ||

||

- **HotspotTopup** || - ||
 - `payment_intent_id`
 - `service_uuid`

- imsi
- 電話番号
- topup_amount
- 電話番号/国/地区
- 国 - 地区
 - 国“地区 - 7 位”
 - 国topup_amount国
 - 国 service_id 国 customer_id
- 国 - 地区
 - 国
 - 国
 - 国Stripe payment_intent_id
- 国 - 地区
 - 国“国 [国]”
 - 国topup_amount国 - 国
 - 国

4. 国

国 playbook play_topup_hotspot 国

- 国 OCS/CGRateS API
- 国
- 国
- 国
- 国

API 国 0.2 国 25 国

5. 国

国

```

if topup_provisioning_failed:
    refund = stripe.Refund.create(
        payment_intent=payment_intent_id,
        reason='requested_by_customer' # 請求された
    )
    status_message = "請求された..."

```

請求された...

API 呼び出し

POST

POST /oam/topup_dongle
Content-Type: application/json

```

{
  "service_uuid": "123e4567-e89b-12d3-a456-426614174000",
  "imsi": "310120123456789",
  "days": 7,
  "payment_intent_id": "pi_1234567890abcdef",
  "topup_amount": 70.00
}

```

レスポンス

```

{
  "result": "OK",
  "status": 200,
  "provision_id": 456,
  "payment_intent_id": "pi_1234567890abcdef",
  "service_uuid": "123e4567-e89b-12d3-a456-426614174000",
  "invoice_id": 789
}

```

成功


```
{
  "result": "Failed",
  "Reason": "OCS connection timeout",
  "status": 500
}
```

□□□□

- □□□□□□□□□□ service_uuid□imsi□days□ payment_intent_id□topup_amount□
- topup_amount □ days □□□ $\text{topup_amount} \times 100 == \text{days} \times 1000$ □□□□□□□
- Stripe □□□□□□□
- □□□□□□□□□ $\text{payment_intent.amount} == \text{topup_amount} \times 100$
- □□□□□□□ `succeeded`
- □□□□□□□□□□□□ HotspotTopup □□

□□□□

□□□□□□□□□□□□□□□□□□

GET /oam/usage

□□□

```
{
  "imsi": "310120123456789",
  "service": {
    "service_uuid": "123e4567-e89b-12d3-a456-426614174000",
    "service_name": "□□□□ - 0412345678",
    "service_status": "□□"
  },
  "balance": {
    "expiry": "2025-01-10T23:59:59Z",
    "unlimited": true
  },
  "requestingIp": "203.0.113.45"
}
```

ローカル環境で IP アドレスを指定する

ローカル

ローカル

OmniCRM-UI の `.env` ファイルを編集する

```
# ローカル環境
VITE_TOPUP_PRICE_PER_DAY=10
VITE_CURRENCY_CODE=AUD
VITE_SELF_CARE_NAME="YourCompany"

# 本番環境
VITE_PAYMENT_VENDORS="stripe"
VITE_STRIPE_PUBLISHABLE_KEY=pk_live_...
```

Stripe 設定

Stripe 設定を行う

1. Stripe アカウントを作成する
2. Webhook 設定を確認する
3. API キーを確認する
4. 環境変数 - 設定を確認する

```
# テスト環境
VITE_STRIPE_PUBLISHABLE_KEY=pk_test_...

# 本番環境
VITE_STRIPE_PUBLISHABLE_KEY=pk_live_...
```

Playbook 実行

playbook `play_topup_hotspot.yaml` を実行する

- `days`
- OCS/CGRateS API
-
-

playbook

```
- name:
  hosts: localhost
  tasks:
    - name: 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 - 2
- Action_Balance_Out -
- Action_Balance_Expired -

□□□□

□□□□□□□

□□ 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 Level 1□
- OmniCRM □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□

0000

- 0000000000000000
- IP 000000000000
- IMSI 00000000000000
- 0000000000

00000

- 000000000000 - 00000000
- 0000000000000000IP 0 IMSI0
- 00000000000000
- 00000000 CRM 000000000000

00000

000000

1. 000000 - 0000000000000000
2. 00000000 - 0000000000
3. 00 **playbooks** 00 - 00 playbooks 000000
4. 00 **OCS** 000 - 00 OCS API 00000
5. 0000 - 000000 `VITE_TOPUP_PRICE_PER_DAY`

00000

1. 0000 **URL** - 00000000
2. 00000000 - SMS 000000
3. 000000000 - 0000000000000000
4. 00000000 - 000000000000

000000

1. 00 **Stripe webhooks** - 00 webhook 00000000000000
2. 00000 - 0000000000 payment_intent_id
3. 0000 - 000000000000000000

4. payments_process - payments_process

5. payments - payments <5 payments

payments

- payments_process - payments
- concepts_provisioning - payments
- payments_process - payments
- payments_transaction - payments
- payments_invoices - payments



OmniCRM

A

Addon -

Ansible - OmniCRM Ansible **Ansible-Based Provisioning**

AuC (Authentication Centre) - SIM OmniCRM AuC SIM HSS

C

CGRateS - OmniCRM

Cell Broadcast - Cell Broadcast System

E

eSIM - SIM OmniCRM eSIM **SIM Card Provisioning**

H

HSS (Home Subscriber Server) - OmniCRM SIM HSS HSS

I

IMS (IP Multimedia Subsystem) - 透過 IP 網路VoLTE提供通話及網路服務
HSS 與 IMS 有密切關係

IMSI (International Mobile Subscriber Identity) - 在 SIM 卡內儲存
在OmniCRM 系統內 IMSI 有儲存

Inventory - 儲存SIM卡資訊
在 [Inventory Management](#)

M

MSISDN (Mobile Station ISDN Number) - 儲存SIM卡 IMSI 號碼
在 MSISDN 系統內??

O

OCS (Online Charging System) - 儲存OCS 資料
OmniCRM 有儲存

P

Playbook - 透過 Ansible 系統 SIM 卡資訊
在 [Ansible Playbooks](#)

PLMN (Public Land Mobile Network) - 儲存 PLMN 資料
在系統內

Product - 儲存 [Products and Services](#)

Proforma Invoice - 儲存
[Invoices](#)

Provisioning - 透過 Ansible 系統
[Provisioning](#)

R

RBAC (Role-Based Access Control) - 在 OmniCRM 中实现基于角色的访问控制
❖❖ 在 OmniCRM 中实现基于角色的访问控制 [Role-Based Access Control](#)

S

Self-Care Portal - 在 OmniCRM 中实现自助服务门户 [Self-Care Portal](#)

Self-Signup - 在 OmniCRM 中实现 SIM 卡自助注册

Service - 在 OmniCRM 中实现产品和服务 [Products and Services](#)

SIM (Subscriber Identity Module) - 在 OmniCRM 中实现 SIM 卡 [SIM Card Provisioning](#)

T

Tag - 在 OmniCRM 中实现标签 [Tags](#)

Transaction - 在 OmniCRM 中实现交易 [Transactions](#)

Numbers

2FA (Two-Factor Authentication) - 在 OmniCRM 中实现双因素认证
在 OmniCRM 中实现双因素认证 [Two-Factor Authentication](#)

OmniCRM 部署方案

OmniCRM 部署方案
Ansible 部署方案

部署环境

OmniCRM 部署环境

1. 部署 - 部署环境
2. 部署 - 部署环境
3. 部署 - 部署环境/部署环境
4. 部署 - 部署环境

部署步骤

OmniCRM 部署步骤

部署

部署环境

- 部署环境
- 部署
- 部署
- 部署

部署

部署环境

部署

部署环境

- `product_name` - 产品名称
- `product_slug` - 产品 API 接口 URL 标识
- `category` - “standalone” 独立产品/服务
- `service_type` - 服务类型
- `enabled` - 是否 true 启用
- `icon` - 产品 UI 图标 FontAwesome 图标
- `comment` - 产品评论

第 3 步

```
{
  "retail_cost": 15.00,
  "wholesale_cost": 5.00,
  "retail_setup_cost": 0.00,
  "wholesale_setup_cost": 1.00,
  "contract_days": 30
}
```

产品

- 零售成本 £15.00
- 批发成本 £5.00
- 零售设置成本 £10.00 200% 零售 67% 零售
- 批发设置成本 -£1.00 批发设置成本
- 合同天数 30 天

第 4 步

```
{
  "residential": true,
  "business": false,
  "customer_can_purchase": true,
  "available_from": "2025-01-01T00:00:00Z",
  "available_until": null
}
```

- 产品

- 10000000000000000
- 1000000
- 2025 1 1 0000
- 10000000000000000

5

```
{
  "auto_renew": "prompt",
  "allow_auto_renew": true
}
```

- "prompt" - 提示词
- "true" - 是
- "false" - 否
- allow_auto_renew: true - 是否自动续费

6



```
{
  "inventory_items_list": ["SIM Card", "Mobile Number"]
}
```

□ □ □ □ □ □ □

OmniCRM

- 00 - 000000000000 “SIM 0” “0000” “00000” 0
- 0000 - 0000MAC 00000000
- 00 - 000000000000
- 00 - 00000000

□ □ □ □ □ □ □ □ □ □

`inventory_items_list` 是 Python 数据类型 `list` 列表

列表

```
# SIM 卡
inventory_items_list: ["SIM Card", "Mobile Number"]

# 调制解调器
inventory_items_list: ["Modem Router", "Static IP Address"]

# 无线路由器
inventory_items_list: []

# CPE
inventory_items_list: ["Fixed Wireless CPE", "IPv4 Address",
                        "IPv6 Prefix"]
```

列表

列表的初始化

1. 初始化

列表的初始化 `inventory_items_list`

2. 列表的初始化

列表的初始化

3. 列表的初始化

列表的初始化

- 列表 - 列表的初始化
- 列表 - `item_state` 列表的初始化
- 列表 - `service_id` 列表的初始化
- 列表 - `customer_id` 列表的初始化

列表的初始化

“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_items_list
```

1. 0000 00 → 00 → 00
2. 00000000000000000000 "SIM 0"0
3. 00000
 - itemtext1_label: "ICCID"
 - itemtext2_label: "IMSI"
 - itemtext3_label: "PUK 0"
4. 00000000000000000000

☐ ☒ ☒ ☒ ☒

□ □ □ □ □ □ □ □

```
inventory_items_list [{"SIM Card", "SIM Card"}]
```

□ □ □ □ □ □ □ □ □ □ □ □ □

00 **1**0000000000000000

```
#  SIM 
inventory_items_list: ["SIM ", "SIM ", ""]
```

□□□□□□□□“□ SIM □”□“□ SIM □”□□□□□□□□□□□□

2

```
# SIM
inventory_items_list: ["SIM", ""]
```

“ SIM ” SIM itemtext1: ICCID itemtext2: ICCID

□ □ □ □ □ □

□□□□□

```
inventory_items_list: ["SIM Card', 'Mobile Number']"
```

- SIM 號碼 ICCID/IMSI 號碼 eSIM
- 國際移動電話號碼MSISDN

無線網路

```
inventory_items_list: ["Modem Router', 'Static IP Address']"
```

- 無線網路設備 MAC 地址 CPE 設備
- 靜態 IP 地址 IPv4

無線網路

```
inventory_items_list: ["Fixed Wireless CPE', 'IPv4 Address',  
'IPv6 Prefix']"
```

- CPE 設備
- IPv4 地址 IP 地址
- IPv6 地址 /56 地址 /64 地址

無線網路設備/無線網路設備

VoIP 設備

```
inventory_items_list: ["DID Number']"
```

- DID 號碼

無線 SIP 設備/無線 SIP 設備**無線設備

GPON/無線

```
inventory_items_list: ["ONT 設備', 'GPON 設備', 'IPv4 設備', '無線設備']"
```

- ONT 管理画面から確認
- GPON 管理画面から OLT 確認
- IPv4 管理画面から IP
- 管理画面から確認

管理画面

```
inventory_items_list: ["'管理画面'"]
```

- 管理画面
- 管理画面

管理画面

1. 管理画面

管理画面

- SIM 管理画面
- IP 管理画面
- 管理画面

管理画面

2. 管理画面

管理画面

- SIM 管理画面
- 管理画面
- 管理画面
- 管理画面

3. 管理画面

管理画面

- SIM 管理画面

- 購買設備
- 購買 CPE 設備
- 購買 IP 設備

4. 設備

設備

- 購買 SIM 設備
- 購買設備
- 購買設備
- 購買 COGS 設備

5. 設備

設備

- 購買 SIM 設備
- 購買設備
- 購買設備
- 購買設備

設備

購買“設備”

購買

- 購買設備
- 購買“設備”
- 購買“設備”
- 購買設備

購買

1. 購買設備 → 購買 → 購買
2. 購買設備
3. 購買設備 → 購買 → 購買

4. 检查是否包含“”“”

5. 检查是否包含 service_id 是否 NULL

检查 检查

检查

- inventory_items_list 是否 "[]"
- inventory_items_list 是否 NULL 是否
- 检查“”“”

检查

- 检查 inventory_items_list: ["'Type1', 'Type2']"
- 检查
- 检查 API 检查 inventory_items_list

检查 “”“”

检查

- 检查
- 检查 ID 检查
- 检查

检查

- 检查 hostvars[inventory_hostname]['SIM Card']
- 检查 {{ hostvars[inventory_hostname]['SIM Card'] | int }}
- 检查

检查 检查

检查 7 检查

检查 检查

```
{
  "features_list": "20GB 記憶體空間00 30 天",
  "terms": "30 天內可隨時取消訂閱"
}
```

記憶體空間

30 天 - 30 天

隨時取消訂閱

1. 記憶體空間 - 記憶體空間
◦ “20GB” “50GB” - 記憶體空間
◦ 隨時取消訂閱
2. 30 天 - 30 天
◦ “30 天” 30 天
◦ 隨時取消訂閱
3. 隨時取消訂閱 - 隨時取消訂閱
◦ 隨時取消訂閱 “隨時取消訂閱” → 隨時取消訂閱
◦ 隨時取消訂閱
◦ 隨時取消訂閱
4. 30 天 - 隨時取消訂閱
◦ 隨時取消訂閱
◦ 隨時取消訂閱
◦ 隨時取消訂閱
5. **SEO** 30 天 - 隨時取消訂閱
◦ 隨時取消訂閱 “隨時取消訂閱” → 隨時取消訂閱
◦ 隨時取消訂閱

30 天 - 30 天

隨時取消訂閱

1. 30 天 - 隨時取消訂閱
◦ “30 天” - 31 天
◦ “隨時取消訂閱” - 隨時取消訂閱

- 國際標準化組織
2. 隱私 - 個人資料
 - “個人資料” - 任何與特定個人有關的信息
 - “匿名化”個人資料 - 無法識別個人
 - 個人資料
3. 隱私 - 個人資料
 - 個人資料保護法
 - 個人資料
 - 個人資料 GDPR/隱私
4. 隱私 - 個人資料
 - “個人資料” - 任何與特定個人有關的信息
 - “匿名化” - 無法識別個人
 - “匿名化” - 無法識別個人
5. 隱私 - 個人資料
 - 個人資料
 - 個人資料
 - 個人資料

5/5

□□□□“□□□□□□”□□□□□□□□□□□□ 10,000 □□□□□□□□□□□□

- [illegible]

[illegible]

- 
- 
- 
- 

□ □ □ □ □ □ □

1. 项目背景

项目背景

2. 项目目标

项目“目标”

- 项目目标
- 项目背景
- 项目背景
- 项目背景
- 项目背景

3. 项目

项目背景

项目 • 20GB 项目 • 项目背景 • 项目背景 • 项目 • 30 项目

项目£15.00/项目 £0.00

[项目] [项目背景]

4. 项目背景

项目背景

项目背景

terms 项目背景

项目 **UI** 项目背景

1. 项目背景

项目背景

2. 项目

项目背景

[00] 0000000000

3. 00

000000000000000000000000

000000

- 000 0000000000 50 00000000000
- 000 0000000000000000000000
- 000 00000000000000000000

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 - 000000000000
- 00000000 0mniCRM-API/Provisioners/plays/play_local_mobile_sim.yaml

00000000

```
{
  "product_name": "SIM 20GB",
  "product_slug": "prepaid-mobile-20gb",
  "category": "standalone",
  "service_type": "mobile",
  "enabled": true,
  "icon": "fa-solid fa-sim-card",
  "comment": "SIM 20GB",

  "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',
    '',
    '',
    '',
    '',
    '30'
  ]",
  "terms": "30",

  "provisioning_play": "play_local_mobile_sim",
  "provisioning_json_vars": "{
    \"days\": 30,
    \"data_gb\": 20,
    \"voice_minutes\": \"unlimited\",
    \"sms_count\": \"unlimited\"
  }
```

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
|--|--|--|--|--|--|--|--|

[illegible]

15GB 内存

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

```

{
  "product_name": "5GB 5GB",
  "product_slug": "5gb-data-boost",
  "category": "addon",
  "service_type": "mobile",
  "enabled": true,
  "icon": "fa-solid fa-plus",
  "comment": "5GB 5GB",

  "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 7 ",
  "terms": "7 ",

  "provisioning_play": "play_topup_charge_then_action",
  "provisioning_json_vars": "{
    \"data_gb\": 5,
    \"days\": 7
  }"
}

```

2

```
{
  "product_name": "WiFi 6 00000000",
  "product_slug": "wifi6-modem-rental",
  "category": "addon",
  "service_type": "internet",
  "enabled": true,
  "icon": "fa-solid fa-router",
  "comment": "WiFi 6 000000000000 - 00",

  "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": "['00000000']",
  "relies_on_list": "",

  "features_list": "WiFi 6 (802.11ax)000 2.4GHz + 5GHz00000 40 00
000000",
  "terms": "000000000000000000000000 £150 000000000000000000",

  "provisioning_play": "play_addon_assign_modem",
  "provisioning_json_vars": "{
    \"device_type\": \"modem_router\",
    \"requires_configuration\": true
  }"
}
```

000000000000

- category: "addon" - 0000000000000000
- contract_days: 000000 30000000 - 0000
- inventory_items_list: "[]" 000000 "['00000000']" 0000 - 0000

- `auto_renew: "false"` - `auto_renew: "true"` - `auto_renew`
- `relies_on_list: ""` - `relies_on_list: "service_type"` - `relies_on_list`

Inventory items list

Inventory items list

1. `auto_renew` - `auto_renew`
2. `relies_on_list` - `relies_on_list`
3. `auto_renew` - `auto_renew`
4. `relies_on_list` - `relies_on_list`
5. `auto_renew` - `auto_renew`
6. `relies_on_list` - `relies_on_list`

Inventory items list

WiFi 6

1. `auto_renew` - `auto_renew`
2. `relies_on_list` - `relies_on_list`
3. `auto_renew` - `auto_renew` £10.00
4. `relies_on_list` - `relies_on_list`
 - `service_id` - `service_id`
 - `customer_id` - `customer_id`
 - `item_state` - `item_state`
5. `auto_renew` - `auto_renew`
6. `relies_on_list` - `relies_on_list` ONT
7. `auto_renew` - `auto_renew` £10

Inventory items list

Inventory items list

1. `auto_renew` - `auto_renew`
2. `relies_on_list`
 - `service_id` - `service_id`
 - `customer_id` - `customer_id`

- 0000000 14 000000

3. 000000

- 000000 item_state = “0000”00000000
- 0 item_state = “00”00000000
- 0000000000000000

4. 00000

- 14 000000 £150 000000
- 000000 item_state = “00”
- 00000000£450+ 00000

0000000000

0000000000000000000000

- 00000000000000000000
- 000000000000000000000000
- 000000000000 contract_days 0000000

00 20000000

000000“000000 20GB”00000OmniCRM 00 Ansible 000000

0000000

00000 → 00000 → 00000000 ↓ ↓ 00000 ← 00000 ← 00 Ansible 00 ↓ ↓ 000000000000 → OCS
00000 → 00000 → 00000

000000000000

1. 0000000

00000000

- 000000“000000”
- 000000000000“000000 20GB”
- 0000000000000000

2. SIM

Information

- **SIM** - SIM card information
 - SIM "SIM-00123 - ICCID: 8944..."
- **Phone** - Phone number
 - Phone "+44 7700 900123"

Information

3. Price

Information

- Price £0.00
- Price £15.00
- Price £15.00
- Price 30

Information

☐ 30

4. API

API

- Provision status=1
- + +
- Ansible
- provision_id UI

5. UI

Information

UI

```
{
  "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

실행 play_local_mobile_sim.yaml 실행

Ansible 실행

실행

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
    # -
    # -
```

環境構築

第 1 章

```
- name: ansible.builtin.include_vars:
  file: ../../crm_config.yaml
  name: crm_config
```

環境構築

- OCS/CGRateS URL 設定
- CRM API URL
- 認証

第 2 章

```
- name: CRM API 設定
  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
```

環境構築

- GET /crm/product/product_id/42
- 認証
- api_response_product 設定

環境構築 provisioning_json_vars 環境構築

- 環境構築
- 環境構築
- 環境構築

- 設定項目を定義

第3ステップ

```
- 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ステップ

```
- 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 IMSI
 - itemtext1 = ICCID (SIM 番号)
 - itemtext2 = IMSI (端末番号)
- 設定項目を定義

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: OCS
  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

```

Annotations

- CGRateS JSON-RPC API
- service_uuid
-
-

Tests

- OCS
-
-

7

```
- name: 20GB
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"
        }
      }]
    }
}
```

配置

- 20GB
- 21474836480 $20 * 1024 * 1024 * 1024$
- 720 30
- 10

8

```
- 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": "手机 - {{ 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

```

返回结果

- 返回结果
- 返回 service_uuid 接口 OCS
- 返回结果
- 返回结果“成功”
- 返回 service_id 接口

返回 10 秒

```
- 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"
    }
```

タスク

- SIM
- service_id SIM
- “”“”
-

タスク

- 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**00000000000000

```

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: 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”
- 部署失败

部署

- OCS 部署
- 部署数据
- 部署失败

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
- “ - +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

📋 需求

👉 20GB 流量 18GB 流量 “5GB 流量” 套餐

1. 套餐

- 套餐 - +44 7700 900123 套餐
- 套餐 20GB 流量 18GB 流量 90%
- 套餐 “套餐” “套餐”

2. 套餐

套餐

- `category = "addon"`
- `service_type = "mobile"` 套餐
- `residential = true` 套餐
- `enabled = true`

套餐 “5GB 流量 - £5.00”

3. 套餐

- 套餐 “5GB 流量”
- 套餐 £5.00
- 套餐

4. 套餐

套餐 `play_topup_charge_then_action.yaml` 套餐

```
{
  "product_id": 43, # 5GB 套餐
  "customer_id": 123,
  "service_id": 1234, # 套餐
  "access_token": "eyJ...",
  "data_gb": 5, # 套餐 provisioning_json_vars
  "days": 7 # 套餐 provisioning_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: 创建支付
  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 商家
- 商家 £5.00
- 商家 £1.50 商家
- 商家
- 商家
- 商家

商家

- 商家
- 商家
- 商家

商家 3 商家 OCS 商家

```
- name: 商家 5GB 商家
  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
          }
        }]
      }
```

商家

- 商家 5GB 5368709120 商家
- 168 商家 7 商家

- 20년째 운영 중인 - 운영 중인 운영

3년 3개월 3개월

- 2GB 25 개
- 5GB 7 개
- 7GB
- 운영 중인 운영

4개

```
- name: 운영 중인
  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 ",
        "description": " 5GB 7 ",
        "retail_cost": 5.00,
        "wholesale_cost": 1.50
      }
}
```

3개

- £5.00
- £1.50
- 운영 중인 운영

3개

1. 운영 중인 운영
2. 운영 중인
3. 운영 중인 OCS

4. 在 CRM 中创建

5. 创建产品时指定 7GB

配置

- 创建产品时 £15 的 £5
- 创建产品时 £5 的 £1.50

创建产品时

创建产品时

配置

```
{
  "product_name": "10GB 产品",
  "category": "addon",
  "retail_cost": 10.00,
  "contract_days": 30,
  "auto_renew": "true",
  "provisioning_play": "play_topup_charge_then_action"
}
```

创建 **ActionPlan**

```

- name: 创建计划 ActionPlan
  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
        }]
      }

```

配置

- OCS 配置
- 30 秒内 `Action_{{ product_slug }}`
- 配置计划
- 配置计划

计划

- 配置计划“计划2025 2 1 - £10.00”
- 配置“计划”计划
- 配置“计划”计划

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

- ...
- “...”
- “...”
- ...

2. 难点

本章难点在于...

- “...”
- 7-30 ...
- ...
- ...

3. 小结

本章小结...


```
{
  "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ActionPlans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Inventory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** - 000000000000
2. **ActionPlans** - 000000
3. **CRM** - 000“000”000000
4. 0000 - SIM 000“000”000000000000

5. □□□□□

□□□□□□□

- 顧客満足度向上
- CRM 活用促進
- 業務効率化
- 情報セキュリティ
- OCS 導入完了

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
|--|--|--|--|--|--|--|--|

□ □ □ □ □ □ □ □ □ □

- 00000000
- OCS 00000000
- 0000
- 000000000000

```
- 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
- OCS 00000000
- 0000
- 00000000

```
- name: 00 OCS 00
  uri:
    url: "http://{{ crm_config.ocs.cgrates }}/jsonrpc"
    method: POST
    body:
      {
        "method": "ApierV2.RemoveAccount",
        "params": [{
          "Account": "{{ service_uuid }}"
        }]
      }
}
```

000000000000

0000000000

00000

- `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“1234”
- 有効期限30 日
- 有効範囲“1234”

設定

設定

- 有効
- 有効期限
- 有効範囲
- 有効範囲

Mailjet ☐ OmniCRM ☐

OmniCRM & Mailjet

11

Mailjet ☐

- **Marketing Automation** - Automatisierung von Marketingprozessen
- **CRM (Customer Relationship Management)** - Verwaltung von Kundenbeziehungen
- **Lead Management** - 10 Schritte zum erfolgreichen Lead Management mit Mailjet
- **CRM Integration** - Integration von CRM mit anderen Systemen (z.B. CSV Import/Export)
- **Reporting & Analytics** - Analyse von Marketingkennzahlen

OmniCRM → Mailjet → Customers → All Contacts → UI

11

Mailjet

□ □ □ □

```
mailjet:
  api_key: your_mailjet_api_key
  api_secret: your mailjet api secret
```

API

1. <<https://www.mailjet.com>>
2. **Account Settings** → **API Keys**
3. **API Key** **Secret Key**

4. `crm_config.yaml`

Configuration

OmniCRM **10** configuration parameters

- **from_email** - email address
- **from_name** - company name
- **template_id** - Mailjet ID
- **subject** - email subject

Example

Configuration

Configuration

```
api_crmCommunicationCustomerWelcome:  
  from_email: "support@yourcompany.com"  
  from_name: "Your Company Support"  
  template_id: 5977509  
  subject: "Welcome to Your Company"
```

Configuration

- Configuration
- Configuration
- Configuration

Configuration

- `{{ var:customer_name }}` - name
- `{{ var:email }}` - email address
- `{{ var:company_name }}` - company name
- `{{ var:login_url }}` - login URL
- `{{ var:support_url }}` - support URL

API 调用

API 调用示例

```
api_crmCommunicationCustomerInvoice:  
  from_email: "billing@yourcompany.com"  
  from_name: "Your Company Billing"  
  template_id: 6759851  
  subject: "Your Invoice - "
```

返回结果

- 返回结果示例
- 返回结果示例
- 返回结果示例

返回结果

- {{ var:customer_name }} - 客户名称
- {{ var:invoice_number }} - 发票 ID/编号
- {{ var:invoice_date }} - 发票日期
- {{ var:due_date }} - 到期日期
- {{ var:total_amount }} - 总金额
- {{ var:invoice_url }} - 发票/账单 PDF 链接
- {{ var:pay_url }} - 付款链接

返回结果

PDF 文件下载链接

返回结果

返回结果


```
api_crmCommunicationCustomerInvoiceReminder:
  from_email: "billing@yourcompany.com"
  from_name: "Your Company Billing"
  template_id: 6759852
  subject: "Payment Reminder - Invoice Overdue"
```

□□□□□

- □□□□ X □□□□□□
- □□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□□

- {{ var:customer_name }}
- {{ var:invoice_number }}
- {{ var:due_date }}
- {{ var:days_overdue }}
- {{ var:total_amount }}
- {{ var:pay_url }}

□□□□□□□□

□□□□□□□□□□□□□□

```
api_crmCommunicationUserWelcome:
  from_email: "admin@yourcompany.com"
  from_name: "Your Company Admin"
  template_id: 5977510
  subject: "Welcome to the Team"
```

□□□□□

- □□□□□□□□□□
- □□□□□□□□“□□□□□□”□□

□□□□□□□

- `{{ var:user_name }}` - ユーザー名
- `{{ var:email }}` - メールアドレス
- `{{ var:role }}` - ロール
- `{{ var:login_url }}` - ログインURL
- `{{ var:temp_password }}` - 仮パスワード
- `{{ var:support_email }}` - IT サポート

メール本文

メール本文のプレビュー

```
api_crmCommunicationUserPasswordReset:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977511
  subject: "Password Reset Request"
```

メール本文

- メール本文のプレビュー
- メール本文のプレビュー

メール本文

- `{{ var:user_name }}`
- `{{ var:reset_url }}` - リセットURL 1 回
- `{{ var:expiry_time }}` - 有効期限

メール本文

メール本文のプレビュー 1 回

メール本文

メール本文のプレビュー

```
api_crmCommunicationUserPasswordResetSuccess:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977512
  subject: "Password Changed Successfully"
```

□□□□□

- □□□□□□□□□□
- □□□□□□□□□□

□□□□□□□

- {{ var:user_name }}
- {{ var:change_date }} - □□□□□□□/□□
- {{ var:ip_address }} - □□□ IP □□□□□□
- {{ var:support_email }} - □□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□

```
api_crmCommunicationUserPasswordChange:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company Security"
  template_id: 5977513
  subject: "Password Change Notification"
```

□□□□□

- □□□□□□□□/□□□□□□□□
- □□□□□□□□□□

□□□□□□□

- {{ var:user_name }}
- {{ var:change_date }}
- {{ var:changed_by }} - “□□”□□□□□□

- `{{ var:support_email }}`

□□□□□□

□□□□□□□□□□□□□□□□

```
api_crmCommunicationEmailVerification:
  from_email: "noreply@yourcompany.com"
  from_name: "Your Company"
  template_id: 5977514
  subject: "Verify Your Email Address"
```

□□□□□□

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

□□□□□□□□

- `{{ var:user_name }}`
- `{{ var:verification_url }}` - □□□□□□□□□□
- `{{ var:verification_code }}` - □□□□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□□□□/□□□□□□□□□□

```
api_crmCommunicationsBalanceExpired:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977515
  subject: "Your Service Balance Has Expired"
```

□□□□□□

- □□□□□□□□
- □□□□□□□□□□□□□□

- 00000000

00000000

- {{ var:customer_name }}
- {{ var:service_name }} - 00000000
- {{ var:expiry_date }}
- {{ var:balance_type }} - “00”“00”“00”
- {{ var:renewal_url }} - 00/000000

000000

00000000000000000000

```
api_crmCommunicationsBalanceLow:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 5977516
  subject: "Low Balance Alert"
```

000000

- 00000000000000 20%
- 000000 OCS 000
- 0000000000

00000000

- {{ var:customer_name }}
- {{ var:service_name }}
- {{ var:current_balance }}
- {{ var:threshold }}
- {{ var:balance_type }}
- {{ var:topup_url }} - 00000000

eSIM 0000

0 eSIM 0000000000000000

```
api_crmCommunicationEsimDelivery:
  from_email: "support@yourcompany.com"
  from_name: "Your Company Support"
  template_id: 0
  subject: "Your eSIM is Ready"
```

📧

- 📧 eSIM
- 📧 eSIM 📧

📧

- `{{ var:customer_name }}` - 📧
- `{{ var:iccid }}` - eSIM ICCID
- `{{ var:activation_code }}` - LPA 📧
- `{{ var:has_qr_code }}` - 📧
- `{{ var:has_activation_code }}` - 📧

📧

📧 `template_id` 📧 0 📧 OmniCRM 📧 HTML 📧 eSIM QR
📧 📧 `crm_config.yaml` 📧 `branding` 📧 📧

📧 Mailjet 📧

📧 Mailjet 📧

📧 1 📧 Mailjet 📧

1. 📧 Mailjet 📧
2. 📧 **Email Templates → Transactional Templates**
3. 📧 **Create a New Template**
4. 📧 **Code your own template** 📧 **Use template builder**

2

Mailjet HTML

- -
- - {{ var:customer_name }} {{ var:user_name }}
- -
- -
- - /
- -

```

<!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>Password Reset Request</h2>

  <p>Hello {{ var:user_name }},</p>

  <p>We received a request to reset your password. Click the
button below to create a new password:</p>

  <a href="{{ var:reset_url }}" class="button">Reset
Password</a>

  <p>This link expires in {{ var:expiry_time }}.</p>

  <p>If you didn't request this, please ignore this email.</p>

  <hr>
  <p style="font-size: 12px; color: #666;">
    Your Company | support@yourcompany.com<br/>
    123 Business St, City, Country
  </p>
</body>
</html>

```

3. Mailjet Template ID

1. Mailjet Template ID
2. Template ID 5977509
3. ID in `crm_config.yaml`

4. Mailjet 設定

1. Mailjet へ **Test Email** を送信
2. メールが送信されることを確認
3. メールが送信されることを確認
4. メールが送信されることを確認

5. OmniCRM のインストール

インストール時に `crm_config.yaml` を

```
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: "Password Reset Request"
```

OmniCRM API をインストール

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

OmniCRM のインストール

OmniCRM のインストールには Mailjet の ESP への登録と CRM のインストールと Mailjet の設定が必要です。

UI

Customers → All Contacts (`/contacts`) に移動します。

`marketing_consent_status` と `customer_status` の **Export CSV** をクリックしてデータをダウンロードします。

API

VIEW_CUSTOMER_CONTACT

| Method | Path | Parameters |
|--------|---------------------|---|
| GET | /crm/contact/search | search, page, per_page, sort, order, filters (JSON) |
| GET | /crm/contact/export | CSV export, max 50,000 records |

CSV fields: contact_id, customer_id, customer_name, customer_status, contact_types, marketing_consent_status, marketing_consent_granted_at, created, last_modified

Filter by marketing_consent_status=granted in CSV export to Mailjet ESP

OmniCRM-API/services/contact_service.py search_contacts, export_contacts_csv

Integration with Mailjet

Mailjet integration with OmniCRM: sending contact data to Mailjet ESP

OmniCRM-API/services/marketing_consent_service.py marketing_consent_status == 'granted'

Enums

Enums

- contact.marketing_consent_status Enum: none | pending | granted | revoked
contact.marketing_consent_granted_at - timestamp

- `marketing_consent` - 一个 Webhook 接收来自 Mailjet 的确认邮件。它包含一个 ID 和 IP 地址，用于验证用户身份。

“确认”由 `crm_config.yaml` 中的 `marketing_consent.consent_text_version` 指定。它决定了邮件的模板。

在

在 `marketing_consent` 中配置 `crm_config.yaml`

```
marketing_consent:
  consent_text_version: "v1-2026-05"
  confirmation_url_path: "/marketing/confirm"
  token_expiry_days: 30
  re_permission_batch_size: 200
  re_permission_rate_per_second: 5
  re_permission_skip_pending_within_hours: 72
```

Mailjet 使用 HTML 模板。

```
mailjet:
  api_crmCommunicationMarketingConfirm:
    from_email: "support@example.com"
    from_name: "Example Mobile Support"
    template_id: 0 # 0 = HTML 模板
    subject: "Please confirm your subscription"
```

Mailjet 模板包含 `firstname` 和 `confirmation_link` 以及 `consent_text_version`。

在 **A** 中

“Example Mobile”是邮件的标题。

1. `routes/auth.py` 中的 `/self-signup-partial` 方法调用 `customer.subscribe_to_newsletter`。

2. `marketing_consent_service.request_consent`
`source='signup'` `IP` `marketing_consent`
3. `marketing_consent` `granted=True` `confirmed_at=NULL` `marketing_consent_status` `pending`
4. Mailjet `{public_url}{confirmation_url_path}?token=...`
5. `GET /crm/marketing-consent/confirm` `confirmed_at` `marketing_consent_status` `granted`

B

“” `/self-care/preferences`

| GET | <code>/crm/marketing-consent/me</code> | + |
|------|--|---|
| POST | <code>/crm/marketing-consent/me/subscribe</code> | |
| POST | <code>/crm/marketing-consent/me/unsubscribe</code> | |

`granted=False` `revoked` OmniCRM

C

`marketing_consent_status='none'`

| Method | Endpoint | Response |
|--------|--|----------------------------|
| GET | /crm/oam/marketing-consent/stats | Summary + Details + Totals |
| POST | /crm/oam/marketing-consent/re-permission?dry_run=true | Summary |
| POST | /crm/oam/marketing-consent/re-permission?dry_run=false | Summary |

Statuses: `none`, `revoked`, `pending`
`re_permission_skip_pending_within_hours` (Optional)
`Closed` (Optional) `request_consent` (Optional)

`re_permission_rate_per_second` (Optional) Mailjet (Optional)
`limit` (Optional)

Implementation

1. `marketing_consent` (Optional)
2. UI **Email Marketing → Marketing Consent** (/email-marketing/marketing-consent) (Optional) `none`
3. **Dry-run re-permission** (Optional)
4. `limit` (Optional) 50 (Optional)
5. `limit` (Optional)
6. **Customers → All Contacts**
`marketing_consent_status = granted` **Export CSV** Mailjet
 ESP

Prometheus

| Label | Value | Description |
|--|--------|--|
| omnicrm_marketing_consent_events_total | event | confirmation_sent, confirmation_failed, revoked, token_redeemed, token_invalid |
| omnicrm_marketing_consent_status_total | status | none, pending, granted, get_global_status |

Mailjet API omnicrm_mailjet_api_* omnicrm_mailjet_emails_sent_*
Monitoring & Metrics

Installation

Prerequisites

- Mailjet API Mailjet ID
- - crm_config.yaml api_key api_secret
 - Mailjet
 - ID Mailjet
 - API

Configuration

- OmniCRM
- -
 - {{ var:variable_name }}
 - OmniCRM API
 - Mailjet

PDF

- 簡便 PDF 管理機能
- 簡便
 - 管理機能
 - 管理機能
 - 簡便 PDF 最大 15MB Mailjet 対応
 - 簡便 PDF 管理

簡便 Mailjet 管理機能

- 簡便 OmniCRM 管理機能 Mailjet
- 簡便
 - 管理機能 granted 管理
 - 管理機能 **Customers → All Contacts** 管理 GET
/crm/contact/export?filters={"marketing_consent_status":["granted"]}
 - 管理機能 UI へ CSV へ Mailjet

管理機能

- administration_configuration - 簡便 Mailjet 管理
- payments_invoices - 管理機能
- authentication_flows - 管理機能
- customer_care - 管理機能
- **Monitoring & Metrics** - Mailjet API 管理機能

管理機能

- Mailjet 管理 <<https://dev.mailjet.com/>>
- Mailjet API 管理 <<https://dev.mailjet.com/email/reference/>>

OmniCRM API

OmniCRM API 提供了一個簡單易用的接口，用於管理您的 CRM 數據。通過 API，您可以輕鬆地獲取、更新和刪除客戶信息、銷售記錄以及市場活動數據。

本文將介紹 API 的基本使用，包括如何獲取 API 令牌、如何構建請求以及如何解析響應。

簡介

OmniCRM API 是一個 RESTful API，使用 JSON 格式進行數據交換。所有請求都必須通過 HTTPS 進行，並且需要包含有效的 API 令牌。

快速入門

- 獲取 API 令牌
- 創建客戶記錄
- 更新客戶信息
- 刪除客戶記錄
- API** 基本用法
- 查詢客戶列表
- 獲取銷售記錄

API 端點

所有請求都應發送到以下端點：

- API 基礎 URL
- API 版本
- API 端點

請求格式：

- 請求頭
- 請求體

- 設定する

設定

- 設定する
- 設定する
- 設定する

設定

- 設定する
- 設定する
- 設定する

設定

API 設定

API 設定

API 設定

API 設定

API 設定 `crm_config.yaml`

```
api_keys:
  your-secure-api-key-minimum-32-characters:
    roles:
      - admin
    description: "Provisioning notification system"
```

API 設定

X-API-KEY API 設定

```
curl -X POST "https://yourcrm.com/crm/notification/send" \
-H "X-API-KEY: your-secure-api-key-minimum-32-characters" \
-H "Content-Type: application/json" \
-d '{
  "user_id": 123,
  "type": "provision_complete",
  "title": "Service Activated",
  "message": "Your mobile service has been successfully
activated",
  "link": "/services/167"
}'
```

Bearer Token

Bearer token

Bearer token API

API

POST /crm/notification/send

API (X-API-KEY) Bearer Token

```
{
  "user_id": 123,
  "type": "provision_complete",
  "title": "Service Activated",
  "message": "Your mobile service has been successfully
activated",
  "link": "/services/167",
  "metadata": {
    "service_id": 167,
    "provision_job_id": "prov-2026-00042",
    "duration_seconds": 45
  },
  "related_ticket_id": 42,
  "related_user_id": 5
}
```

欄位

| 欄 | 欄 | 欄 | 欄 |
|-------------------|----|---|----------------|
| user_id | 欄 | 欄 | 欄欄欄欄 ID |
| type | 欄欄 | 欄 | 欄欄欄欄欄欄欄欄 |
| title | 欄欄 | 欄 | 欄欄欄欄 255 欄欄欄 |
| message | 欄欄 | 欄 | 欄欄欄欄 |
| link | 欄欄 | 欄 | 欄欄欄欄 URL/欄 |
| metadata | 欄 | 欄 | 欄欄欄欄欄欄JSON欄 |
| related_ticket_id | 欄 | 欄 | 欄欄欄 ID |
| related_user_id | 欄 | 欄 | 欄欄欄 ID欄欄欄欄欄欄欄欄 |

欄位

| 項目 | 項目 | 説明 |
|--------------------|----------------|-----------------------|
| provision_complete | プロビジョニング完了 | Ansible playbook 実行完了 |
| provision_failed | プロビジョニング失敗 | プロビジョニング失敗 |
| ticket_mention | チケットメンション @ユーザ | 通知 |
| ticket_assigned | チケット割り当て | 通知 |
| ticket_updated | チケット更新 | チケット更新通知 |
| customer_reply | 顧客返信 | 通知 |
| system_alert | システムアラート | システムアラート |
| billing_invoice | 請求書 | 請求書 |
| billing_payment | 請求書 | 請求書 webhook |
| service_expiring | サービス有効期限切れ | 通知 |
| custom | カスタム | カスタム |

通知

通知 201 Created

```
{
  "result": "Success",
  "message": "Notification sent successfully",
  "notification": {
    "notification_id": 1234,
    "user_id": 123,
    "type": "provision_complete",
    "title": "Service Activated",
    "message": "Your mobile service has been successfully
activated",
    "link": "/services/167",
    "metadata": {
      "service_id": 167,
      "provision_job_id": "prov-2026-00042",
      "duration_seconds": 45
    },
    "is_read": false,
    "created_at": "2026-01-11T14:30:00Z"
  }
}
```

□□□□

□□□□□□ (400 Bad Request)□

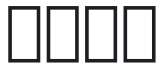
```
{
  "result": "Failed",
  "reason": "user_id is required"
}
```

□□□□□ (404 Not Found)□

```
{
  "result": "Failed",
  "reason": "User 999 not found"
}
```

□□□□□□ (401 Unauthorized)□

```
{  
  "message": "Invalid API key"  
}
```



Python ☐☐

```
import requests

# ☐☐
CRM_URL = 'https://yourcrm.com'
API_KEY = 'your-secure-api-key-minimum-32-characters'

def send_notification(user_id, notification_type, title,
message=None, link=None, metadata=None):
    """☐☐ OmniCRM API ☐☐☐☐☐☐"""

    headers = {
        "Content-Type": "application/json",
        "X-API-KEY": API_KEY
    }

    payload = {
        "user_id": user_id,
        "type": notification_type,
        "title": title
    }

    # ☐☐☐☐☐☐
    if message:
        payload["message"] = message
    if link:
        payload["link"] = link
    if metadata:
        payload["metadata"] = metadata

    response = requests.post(
        f"{CRM_URL}/crm/notification/send",
        headers=headers,
        json=payload
    )

    if response.status_code == 201:
        print(f"✓ ☐☐☐☐☐☐: {response.json()}")
        return response.json()
```

```

else:
    print(f"x {response.text}")
    response.raise_for_status()

# 测试
if __name__ == "__main__":
    # 发送通知
    send_notification(
        user_id=123,
        notification_type="provision_complete",
        title="测试通知",
        message=f"ID 167 测试通知",
        link="/services/167",
        metadata={
            "service_id": 167,
            "provision_job_id": "prov-2026-00042",
            "duration_seconds": 45,
            "playbook": "mobile_service_activation"
        }
    )

```


Bash/cURL ☐☐

```
#!/bin/bash
```

```
# ☐☐
```

```
CRM_URL="https://yourcrm.com"
```

```
API_KEY="your-secure-api-key-minimum-32-characters"
```

```
# ☐☐☐☐
```

```
curl -X POST "${CRM_URL}/crm/notification/send" \
```

```
-H "X-API-KEY: ${API_KEY}" \
```

```
-H "Content-Type: application/json" \
```

```
-d '{
```

```
  "user_id": 123,
```

```
  "type": "provision_complete",
```

```
  "title": "Service Activated",
```

```
  "message": "Your mobile service has been successfully  
activated",
```

```
  "link": "/services/167",
```

```
  "metadata": {
```

```
    "service_id": 167,
```

```
    "provision_job_id": "prov-2026-00042",
```

```
    "duration_seconds": 45
```

```
  }
```

```
}'
```

Node.js ☐☐

```
const axios = require('axios');

const CRM_URL = 'https://yourcrm.com';
const API_KEY = 'your-secure-api-key-minimum-32-characters';

async function sendNotification(userId, type, title, options = {})
{
  try {
    const response = await axios.post(
      `${CRM_URL}/crm/notification/send`,
      {
        user_id: userId,
        type: type,
        title: title,
        message: options.message,
        link: options.link,
        metadata: options.metadata,
        related_ticket_id: options.relatedTicketId,
        related_user_id: options.relatedUserId
      },
      {
        headers: {
          'X-API-KEY': API_KEY,
          'Content-Type': 'application/json'
        }
      }
    );

    console.log('✓ ☐☐☐☐:', response.data);
    return response.data;
  } catch (error) {
    console.error('✗ ☐☐☐☐:', error.response?.data ||
error.message);
    throw error;
  }
}

// ☐☐☐☐
sendNotification(
  123,
  'provision_complete',
```

```
'00000',
{
  message: '000000000000',
  link: '/services/167',
  metadata: {
    service_id: 167,
    provision_job_id: 'prov-2026-00042',
    duration_seconds: 45
  }
}
);
```

0000

Ansible 00 Playbook

0 Ansible playbook 00000000

```

# Ansible Playbook
import requests

def notify_provisioning_complete(service_id, user_id,
success=True):
    """

    notification_type = "provision_complete" if success else
    "provision_failed"
    title = " " if success else " "
    message = f" {service_id} { ' ' if success else ' '
    '}'"

    requests.post(
        "https://yourcrm.com/crm/notification/send",
        headers={"X-API-KEY": "your-api-key"},
        json={
            "user_id": user_id,
            "type": notification_type,
            "title": title,
            "message": message,
            "link": f"/services/{service_id}",
            "metadata": {
                "service_id": service_id,
                "playbook": "mobile_service_activation",
                "success": success
            }
        }
    )

```

```

def notify_invoice_generated(customer_id, invoice_id, amount,
due_date):
    """通知メールを送信"""

    # 顧客のIDを取得
    user_id = get_customer_primary_contact_user_id(customer_id)

    if user_id:
        requests.post(
            "https://yourcrm.com/crm/notification/send",
            headers={"X-API-KEY": "your-billing-api-key"},
            json={
                "user_id": user_id,
                "type": "billing_invoice",
                "title": f"通知 #{invoice_id}",
                "message": f"お支払いの金額は ${amount} 円です。支払期限は {due_date} です",
                "link": f"/invoices/{invoice_id}",
                "metadata": {
                    "invoice_id": invoice_id,
                    "amount": amount,
                    "due_date": due_date,
                    "customer_id": customer_id
                }
            }
        )

```



通知メールを送信

```

def send_monitoring_alert(admin_user_ids, alert_title,
    alert_message, severity="warning"):
    """ """

    for user_id in admin_user_ids:
        requests.post(
            "https://yourcrm.com/crm/notification/send",
            headers={"X-API-KEY": "your-monitoring-api-key"},
            json={
                "user_id": user_id,
                "type": "system_alert",
                "title": alert_title,
                "message": alert_message,
                "metadata": {
                    "severity": severity,
                    "alert_time": datetime.now().isoformat(),
                    "source": "prometheus_alertmanager"
                }
            }
        )

# """
send_monitoring_alert(
    admin_user_ids=[1, 2, 3],
    alert_title="△ ",
    alert_message=" 85%",
    severity="warning"
)

```

Webhook

```

from flask import Flask, request

app = Flask(__name__)

@app.route('/webhook/payment', methods=['POST'])
def payment_webhook():
    """Payment webhook"""

    payment_data = request.json

    if payment_data['status'] == 'succeeded':
        # Send notification
        requests.post(
            "https://yourcrm.com/crm/notification/send",
            headers={"X-API-KEY": "your-payment-webhook-api-key"},
            json={
                "user_id": payment_data['user_id'],
                "type": "billing_payment",
                "title": "Payment received",
                "message": f"Payment of ${payment_data['amount']} received",
                "link": f"/payments/{payment_data['payment_id']}",
                "metadata": {
                    "payment_id": payment_data['payment_id'],
                    "amount": payment_data['amount'],
                    "payment_method":
payment_data['payment_method']
                }
            }
        )

    return {"status": "ok"}

```

Payment received

Payment of \$100.00 received

```

import datetime
from sqlalchemy import create_engine, text

def check_expiring_services():
    """
    检查 7 天内即将过期的服务
    """

    engine = create_engine("postgresql://...")

    # 查询语句
    query = text("""
        SELECT s.service_id, s.end_date, u.id as user_id,
        s.service_name
        FROM services s
        JOIN customer_contacts cc ON s.customer_id =
        cc.customer_id
        JOIN users u ON cc.contact_id = u.contact_id
        WHERE s.end_date BETWEEN CURRENT_DATE AND CURRENT_DATE +
        INTERVAL '7 days'
        AND s.deleted = false
    """)

    with engine.connect() as conn:
        results = conn.execute(query)

        for row in results:
            days_until_expiry = (row.end_date -
            datetime.date.today()).days

            requests.post(
                "https://yourcrm.com/crm/notification/send",
                headers={"X-API-KEY": "your-automation-api-key"},
                json={
                    "user_id": row.user_id,
                    "type": "service_expiring",
                    "title": f"您的服务 {days_until_expiry} 天后到期",
                    "message": f"您的服务 {row.service_name} 将在 {row.end_date} 到期",
                    "link": f"/services/{row.service_id}",
                    "metadata": {
                        "service_id": row.service_id,
                        "days_remaining": days_until_expiry,
                        "expiry_date": str(row.end_date)
                    }
                })

```



```

    }
)

# 每隔 cron 秒
if __name__ == "__main__":
    check_expiring_services()

```

部署

API 部署

1. 生成随机字符串

```
openssl rand -base64 48
```

2. 部署到云服务器并安装 API 服务
3. 部署到云服务器 `crm_config.yaml` 文件
4. 部署到云服务器 API 服务
5. 部署到云服务器 API 服务
6. 部署到云服务器 API 服务

部署

1. 部署到云服务器并安装“部署”服务
2. 部署到云服务器
3. 部署到云服务器
4. 部署到云服务器
5. 部署到云服务器
6. 部署到云服务器

準備

1. 環境構築
2. 設定
3. 実行
4. 結果確認 **ID**
5. **API** の確認

実行

1. 実行
2. 結果確認
3. 結果確認 **API** の確認
4. 結果確認 **HTTP** の確認

結果

API の確認

実行結果の **API** の確認

```
# 実行結果
curl -X POST "https://yourcrm.com/crm/notification/send" \
-H "X-API-KEY: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "user_id": 1,
  "type": "custom",
  "title": "実行結果",
  "message": "実行結果の API の確認"
}'
```

201 Created

ID

```
# 
response = send_notification(
    user_id=1, # 
    notification_type="custom",
    title="",
    message=" API "
)

print(f" ID: {response['notification']['notification_id']}")
```

OmniCRM

401 Unauthorized

- API
- `crm_config.yaml` API
- API 32

404

- `user_id`
-
- `SELECT id FROM users WHERE id = ?`

400 Bad Request - "user_id is required"

-
- `user_id``type``title`
- JSON

データベース

- 通知の取得
- 通知の送信
- 通知の取得 (SQL)

```
SELECT * FROM notifications WHERE user_id = ? ORDER BY created_at DESC
```

500 Internal Server Error

- 通知の取得
- 通知の送信 CRM の通知機能
- 通知の送信 `/var/log/omnicrm/api.log` を確認

API 一覧

API

| メソッド | URL | レスポンス | 認証 |
|--------|---|----------|------------------|
| POST | <code>/crm/notification/send</code> | 通知の送信結果 | API 認証
Bearer |
| GET | <code>/crm/notification/</code> | 通知の一覧 | Bearer |
| GET | <code>/crm/notification/unread-count</code> | 未読通知の数 | Bearer |
| POST | <code>/crm/notification/{id}/mark-read</code> | 通知の既読処理 | Bearer |
| POST | <code>/crm/notification/mark-all-read</code> | 全通知の既読処理 | Bearer |
| DELETE | <code>/crm/notification/{id}</code> | 通知の削除 | Bearer |

📌

- 📌 API 📌
- 📌
- 📌

📌

- 📌 API 📌
- 📌 @📌
- 📌Ansible 📌
- 📌
- RBAC📌



OmniCRM 50

OmniCRM

OmniCRM API を Prometheus で監視するための設定を説明します。
Prometheus は `/crm/metrics` エンドポイントからデータを収集します。

前提条件

必要な環境:

- Prometheus - 監視システム
- Grafana - 可視化ツール
- API - OCS/CGRateS, Stripe, Mailjet API など
- 認証 - API キー/トークン
- **HTTP** プロキシ - API への接続を管理

設定

URL: `http://your-omnicrm-api:5000/crm/metrics`

例: Prometheus の設定

例: Prometheus の監視ジョブ定義

インストール

1. Ansible

Ansible を使用して Prometheus の監視ジョブを自動的にインストール/更新するための **Provisioning System** を **Ansible Playbooks** で定義します。

omnicrm_provision_jobs_total

例: 監視ジョブ

📄:

- `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` - 错误类型: `fatal`, `task_failed`, `timeout`
- `playbook` - Ansible 剧本

注意: 配置项名称与变量名称一致

2. 配置项

本文档描述了OmniCRM 的 SQLAlchemy 数据库配置项。请参考 [System Architecture](#)

omnicrm_db_query_duration_seconds

类型: 浮点

默认:

- `operation` - SQL 操作: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- `table` - 表名

范围: [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0]

描述: 配置 SQLAlchemy 数据库查询超时时间

omnicrm_db_pool_size

类型: 整数

默认: 数据库连接池大小

omnicrm_db_pool_overflow

类型: 整数

默认: 数据库连接池溢出时的行为

omnicrm_db_pool_connections_checked_out

类型: 整数

默认: 数据库连接池检查出的连接数

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` など

単位: 個

単位: 個

3. OCS/CGRateS 連携

OCS連携に関するCGRateS 連携に関する [Billing Overview](#) と [CGRateS Integration](#)

omnicrm_ocs_api_calls_total

単位: 個

項目:

- `method` - OCS API 名: `GetBalance`, `SetBalance`, `SetAccount`, `RemoveAccount` など
- `status` - ステータス: `success`, `failed`

単位: OCS/CGRateS API 呼び出し回数 API 名 OCS 連携

omnicrm_ocs_api_duration_seconds

単位: 秒

項目:

- `method` - OCS API 名: `GetBalance`, `SetBalance`, `GetAccount` など

単位: `[0.1, 0.25, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0]` 秒

omnicrm_ocs_api_errors_total

☐: OCS/CGRateS API ☐☐☐☐☐☐ OCS API ☐☐☐☒☒☐☐☐☐☐

omnicrm_ocs_action_plan_operations_total

CGRateS

Stripe Payment System Guide

omnicrm_stripe_api_calls_total

type: string

type:

- operation - Stripe operation: create_customer, charge, refund, update_payment_method
- status - status: success, failed

type: Stripe API operation duration in seconds

omnicrm_stripe_api_duration_seconds

type: string

type:

- operation - Stripe operation

type: [0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0]

type: Stripe API operation errors total

omnicrm_stripe_api_errors_total

type: string

type:

- operation - Stripe operation
- error_type - error type: card_declined, network_error, api_error

type: Stripe API operation payment amount in cents

omnicrm_stripe_payment_amount_cents

type: string

type:

- payment_type - payment type: charge, refund

例: Stripe 失敗の総数

omnicrm_stripe_payment_failures_total

例: 100

例:

- reason - 理由: card_declined, insufficient_funds, expired_card

例: Stripe 失敗の総数

5. Mailjet 例

Mailjet 例 [Mailjet Integration](#)

omnicrm_mailjet_api_calls_total

例: 100

例:

- email_type - メール種別: welcome, user_welcome, invoice, notification
- status - ステータス: success, failed

例: Mailjet API 呼び出し @track_mailjet_call

omnicrm_mailjet_api_duration_seconds

例: 100

例:

- email_type - メール種別

例: [0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0]

例: Mailjet API 呼び出し 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` - 値: `playbook_single`, `playbook_chain`, `periodic_task`

例: 背景ジョブタイマー `BackgroundJobTimer` の初期化

`omnicrm_background_jobs_active`

例: 有効

例:

- `job_type` - 値

例: 背景ジョブタイマーの初期化

`omnicrm_background_job_duration_seconds`

例: 有効

例:

- `job_type` - 値
- `status` - 値: `success`, `failed`

例: `[1, 5, 10, 30, 60, 120, 180, 300, 600, 1800, 3600]` 1分1秒

例: 背景ジョブタイマーの初期化

`omnicrm_background_job_errors_total`

例: 有効

例:

- `job_type` - 値
- `error_type` - 値

例: 背景ジョブタイマーの初期化

7. Flask HTTP 메트릭들

이 섹션에서는 `prometheus-flask-exporter`가 제공하는 API 및 HTTP 메트릭

`flask_http_request_duration_seconds`

타입: 가우시안

메트릭:

- `method` - HTTP 메서드: `GET`, `POST`, `PUT`, `DELETE` 등
- `endpoint` - Flask 엔드포인트
- `status` - HTTP 상태

타입: API 및 HTTP 메트릭들

`flask_http_request_total`

타입: 가우시안

메트릭:

- `method` - HTTP 메서드
- `endpoint` - Flask 엔드포인트
- `status` - HTTP 상태

타입: API 및 HTTP 메트릭들

`flask_http_request_exceptions_total`

타입: 가우시안

메트릭:

- `method` - HTTP 메서드
- `endpoint` - Flask 엔드포인트

타입: HTTP 메트릭들

8. API 錯誤訊息

OmniCRM API 錯誤訊息

omnicrm_api_errors_total

單位: 個

欄位:

- `endpoint` - API 端點
- `error_type` - 錯誤類型
- `status_code` - HTTP 狀態碼

單位: 個

omnicrm_api_auth_failures_total

單位: 個

欄位:

- `auth_method` - 認證方法: `jwt`, `api_key`, `ip_whitelist`
- `reason` - 原因

單位: 個

9. 環境變數

omnicrm_api

單位: 個

欄位:

- `version` - API 版本
- `environment` - 環境: `production`, `staging`, `development`

單位: OmniCRM API 環境變數

□□□□

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. 監視指標 - 監視 KPI

- Stripe 監視
- 監視
- 監視

3. 監視 - 監視

- API 監視
- 監視
- API 監視

4. 監視 - 監視

- 監視
- 監視
- API 監視

監視

Prometheus 監視

- 15 - 監視15 監視
- 90 - 監視5 監視
- 1 - 監視1 監視

Thanos Cortex VictoriaMetrics 監視

監視

監視

監視 /metrics 監視 nginx 監視

監視

```
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

- 1

:

- 5
 - 5
 - OCS/CGRateS: 5
 - Stripe: 5
 - Mailjet: 5
 - : 4
 - HTTP/Flask: 3
 - : 1
 - : 2
-

- [System Architecture](#) -
- [Provisioning System](#) -
- [Ansible Playbooks](#) -
- [API Documentation](#) - API
- [Billing Overview](#) -
- [Payment System Guide](#) - Stripe
- [Mailjet Integration](#) -
- [Administration Configuration](#) -

OmniCRM のインストールとアップグレードのガイドは、[Stripe](#) と [PayPal](#) のインストールとアップグレードのガイドを参照してください。

OmniCRM のインストールとアップグレードのガイドは、[OmniCRM](#) のインストールとアップグレードのガイドを参照してください。

- stripe paypal
- Visa Mastercard
-
- /

- 000000/00
- 00000000000000

00000000

000000

1. 000 00 → **[00000]**
2. 00 00 00
3. 000 0000 00

0000

000000000000

00000000000000

00000000000000 60 0000000000

00000000

0000000000000000

0000

- 名称 - 必ずしも「株式会社」や「株式会社 Visa」
- 名称 - 必ずしも「株式会社」
- 名称 - 必ずしも「株式会社MM/YYYY 株式会社」
- 名称 - 必ずしも「株式会社」
- 名称 - 必ずしも「株式会社株式会社株式会社」

株式会社

株式会社株式会社 : 株式会社

- 名称 - 株式会社株式会社
- 名称 - 株式会社

株式会社

株式会社 “株式会社” 株式会社株式会社

株式会社 **1**株式会社株式会社

株式会社株式会社 Stripe Elements 株式会社 PayPal SDK 株式会社

株式会社

- 名称 - 株式会社CVC株式会社 Stripe 株式会社

- 銀行口座 - 振込先
- 住所/会社 - 請求先

お問い合わせ

- お問い合わせ - お問い合わせ先

お問い合わせ

- 決済方法: 銀行振込 / クレジットカード / 銀行口座 / iframe 決済 / Stripe Elements / PayPal SDK
- OmniCRM 連携 / 連携先
- PCI DSS 対応 / 対応先
- 決済手数料 / 手数料

お問い合わせ 2

お問い合わせ “お問い合わせ”

1. お問い合わせ
 - お問い合わせ先
 - お問い合わせ先
 - お問い合わせ CVC 番号
2. お問い合わせ
 - お問い合わせ先 OmniCRM
 - お問い合わせ先 pm_1A2B3C4D Stripe
 - お問い合わせ OmniCRM
3. お問い合わせ
 - OmniCRM お問い合わせ先
 - お問い合わせ先
 - お問い合わせ OmniCRM

お問い合わせ 3

お問い合わせ

お問い合わせ Visa 1234

□□□□□□□□□□

□□□□□□

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

□□□□□□□□

□□□□□□□□

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

□□□□□□□□

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

Visa □□ 5678 □□□□□□□□□□□□

□□□□□□□□□□□□□□

□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□□

API 入門

APIとは何か？

APIとは？ 1. APIの定義

- APIは、異なるソフトウェアコンポーネント間でデータを交換するためのインターフェースである。
- APIは、特定のタスクを実行するための一連のルールと手順を提供する。

APIは、ソフトウェア開発者之间的桥梁であり、異なるシステムやサービスが互いに連携できるようにする。

APIは、特定のタスクを実行するための一連のルールと手順を提供する。

APIは、ソフトウェア開発者之间的桥梁であり、異なるシステムやサービスが互いに連携できるようにする。

API 2. APIの種類

APIは、その用途や提供方法によって分類される。

APIの種類

[API] [APIの種類]

APIは、ソフトウェア開発者之间的桥梁であり、異なるシステムやサービスが互いに連携できるようにする。

API “APIの種類” APIの種類

API 3. APIの設計

APIの設計

APIの設計は、APIの目的と用途に基づいて行われる。

APIの設計

- APIの設計は、APIの目的と用途に基づいて行われる。
- APIの設計は、APIの目的と用途に基づいて行われる。
- APIの設計は、APIの目的と用途に基づいて行われる。

XXXXXXXXXX

OmniCRM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX

XXXX XX → XXXXXX XXXXXXXXXXXXXXXX

XX XX XXXX XXXX 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 - XXXX 30/60/90 XXXX
- XXXX - XXXX
- XXXX - XXXXXXXX

XX

- XX - XXXXXXXXXXXXXXXX
- XX - XXXXXXXXXXXXXXXX Mailjet

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

Mailjet

- 60 秒 - 1 分鐘
- 30 秒 - 半分鐘
- 7 秒 - 幾秒鐘
- - 幾分鐘

□ □

□□□□□□□□

Mailjet ☐

- 
- 
- 
- 

```
integrations_mailjet
```

□ □ ◆ ? ◆ ? ◆ □ □ □ □ □ □

□□□□□□□

1. □□□□□□□□□□
2. □□□□□□□□□□
3. □□□ □□ → □□□□
4. □□ “□□□□□□”
5. □□□□□□□□□□□□□□□□□□□□□□□□□□□□
6. □□□□□□□□
7. □□□□/□□□□

□□□□□□□

□□□□□□□□□□

1. 目標を定める
2. 現状を把握 → 課題を抽出
3. 課題を解決するための施策を立案する
4. 施策を実行する
5. 進捗を確認する
6. 振り返り



注意

施策立案の際には、以下の点に注意する必要があります。

- 施策の目的が明確であること
- 施策の実行が可能なこと
- 施策の効果が測定可能なこと

施策立案のステップ

現状把握と課題抽出

施策立案

1. 現状把握
 - 現状の把握（現状の把握）
 - 課題の抽出（課題の抽出）
 - 課題の整理（課題の整理）
 - 課題の優先順位付け（課題の優先順位付け）
2. 施策立案
 - 施策の立案（施策の立案）
 - 施策の実行（施策の実行）
 - 施策の評価（施策の評価）
3. 振り返り
 - 振り返り - 振り返り（振り返り）
 - 振り返り - 振り返り（振り返り）
 - 振り返り - 振り返り（振り返り）

振り返り

□ 1-3 □□□□□□

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

□ 4-7 □□

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

□ 8-14 □□

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

□ 15 □□□□□□

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

□□□□□□□□

□□□□□□□□□□

- □□□□ **30** □ □□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□

1. □□□□□□□□□□□□

2. 商家
3. 商家支付方法
4. 商家支付方法
5. 商家支付方法

商家支付方法

商家

OmniCRM 商家支付方法

1. 商家 → 商家支付方法
2. 商家支付方法 → 商家
3. 商家 **OmniCRM** → 商家支付方法
4. 商家 → 商家支付方法

OmniCRM 商家支付方法

```
{
  "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 商家支付方法

- 商家
- CVV/CVC 碼
- 商家
- PIN 碼

PCI 測試

測試環境與測試步驟

- 測試 **PCI** 測試 - 測試環境 OmniCRM 測試
- 測試環境 - 測試環境測試環境 iframe 測試
- 測試 - 測試環境測試環境
- 測試 - 測試環境 HTTPS/TLS

測試 測試環境 測試環境測試環境

測試環境

測試環境 1 測試環境測試環境

測試 測試環境

1. 測試環境
2. 測試環境
3. 測試環境
4. 測試 Stripe 測試環境
5. 測試環境
6. 測試環境
7. 測試環境
8. 測試環境

測試環境 2 測試環境測試環境

測試 測試環境

1. 測試環境測試環境 60 測試
2. 測試環境
3. 測試 測試 → 測試
4. 測試環境 12/2025
5. 測試 “測試環境”

6. 到期日 12/2028 到期
7. 到期地點
8. 到期
9. 到期地點

到期 3 個月內到期地點

到期 3 個月內到期地點

1. 到期地點
2. 到期地點
3. 到期地點 01/2025
4. 到期“到期地點”
5. 到期地點
6. 到期地點 → 【到期】 → 到期
7. 到期 “到期地點”
8. 到期地點
9. 到期地點
10. 到期地點
11. 到期地點
12. 到期地點“到期地點”

到期 4 個月內到期地點

到期 4 個月內到期地點

1.  到期地點 Visa 到期 1111
2. 到期地點
3. 到期地點 Mastercard 到期 2222 到期地點
4. 到期地點 Amex 到期 3333 到期地點
5. 到期地點 Mastercard
6. 到期 Visa 到期地點

📧 5 個のステップでメールマーケティング

📌 準備段階

1. 目標を設定 → 何を達成したいか
2. ターゲットを特定 (30 代)
3. 15 分以内のメールを作成
4. 送信 → “送信” ボタンをクリック
5. Mailjet を利用してメールを送信
6. 送信結果を確認
7. 返信メールを送信
8. 開封率を確認
9. 開封率を確認

📧 送信

“送信” ボタンをクリック

- 送信 ボタンをクリックすると、メールが送信されます
- 送信結果
- 送信成功
- 送信失敗
- 送信エラー
- 送信エラー

“送信” ボタンをクリック

- 送信 ボタンをクリック API を利用して
- 送信結果
- 送信成功
- 送信失敗
- 送信エラー
- 送信エラー
- 送信エラー

プラットフォームの構築

- 開発 環境の構築
- 認証
 - 認証サーバーの構築
 - 認証クライアントの構築

プラットフォームの“拡張”

- 開発 環境の構築
- 認証
 - 認証
 - 認証サーバーの構築
 - 認証クライアントの構築

プラットフォーム

- 開発 環境の構築
- 認証
 - 認証サーバー
 - 認証クライアント
 - 認証クライアント

プラットフォーム

- 開発 環境の構築 SDK API 環境の構築
- 認証
 - 認証サーバー
 - 認証クライアント
 - 認証クライアント
 - 開発 SDK 環境の構築
 - 認証クライアント/認証クライアント

プラットフォーム

-  Mailjet 環境の構築
- 認証

- 編輯 *crm_config.yaml* 編輯 Mailjet 編輯
- 測試發送郵件
- 測試發送短信
- 編輯 Mailjet 測試發送郵件

測試發送

測試發送

- 測試發送郵件
- 測試 2 測試發送郵件
- 測試 30 測試發送郵件
- 測試/測試發送郵件
- 測試發送“*Visa*”“*Amex*”
- 測試發送郵件

測試發送

- 測試發送郵件
- 測試發送郵件/SMS/測試發送郵件
- 測試發送郵件
- 測試發送郵件
- 測試發送郵件
- 測試發送郵件 £0.01 測試

測試發送

- 測試發送郵件
- 測試 60/30/7 測試發送郵件
- 測試發送郵件/測試發送郵件
- 編輯 Mailjet 測試發送郵件
- 測試發送郵件
- 測試發送郵件  測試

□□□□

- □□□□□ - □□□□ API □□□□
- payments_process - □□□□□□□□□□
- payments_invoices - □□□□□□□□□□
- features_topup_recharge - □□□□□□□□□□
- basics_payment - □□□□□□□□
- customer_care - □□□□□□□□□□□□

OmniCRM 開箱 API 開箱

簡介

OmniCRM 是一個整合了多種支付方式的 CRM 系統，支援 Stripe、PayPal、Square、Adyen、Braintree 等支付平台。

本文將介紹如何透過 API 進行整合。

本文將使用 **Ansible Playbook** 來配置 API 整合。Playbook 是一個 Ansible 的配置文件，用於定義任務和變量。本文將使用 **Playbooks** 來配置 API 整合。

步驟

1. 安裝 Ansible
2. 配置 Ansible
3. 安裝 API 客戶端
4. 配置 API 客戶端
5. 安裝 API 客戶端
6. 配置 API 客戶端
7. API 客戶端
8. 測試 API 客戶端

安裝 Ansible

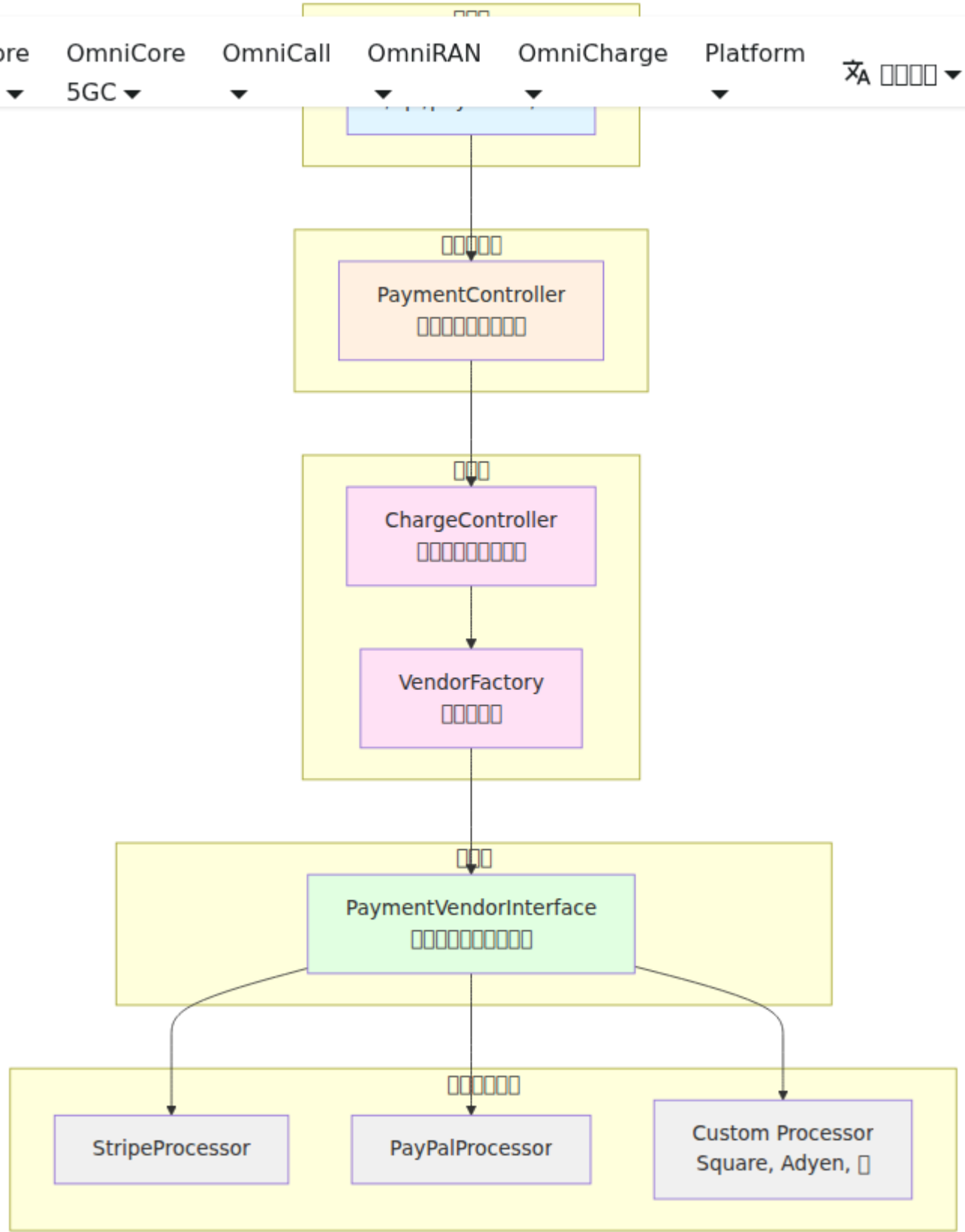
安裝 Ansible

Ansible 是一個基於 Python 的自動化配置管理工具，用於配置管理、應用部署、任務執行等。本文將介紹如何安裝 Ansible。

- 安裝 Ansible 客戶端
- 安裝 Ansible 服務端

- 支付网关
- 支付网关API

支付网关



Payment Vendor

Payment Vendor: Square, Adyen, Braintree, OmniCRM, ...

Payment Vendor: Stripe, PayPal, ...

Payment Vendor



Payment Vendor

1. Create Payment Vendor Interface
2. Create **VendorFactory**
3. Create Payment Vendor
4. Create - Create API

Payment Vendor

```
# Create Square Payment Vendor
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"
  }'
```

Payment Vendor API

Payment Vendor 1-2

```
stripe:
  secret_key: 'sk_live_YOUR_SECRET_KEY_HERE'
  publishable_key: 'pk_live_YOUR_PUBLISHABLE_KEY_HERE'
  currency: 'aud'
  statement_descriptor_suffix: 'YOUR_COMPANY'
```

API キーを `crm_config.yaml` に Stripe から取得した API キー

3. 環境

フロント UI の `.env` ファイル

```
VITE_STRIPE_PUBLISHABLE_KEY=pk_live_YOUR_PUBLISHABLE_KEY_HERE
```

環境変数の設定が完了しました

PayPal 設定

PayPal 開発者コンソールから PayPal API キー

1. PayPal API キー

- <https://developer.paypal.com/dashboard/> から REST API キー
- 開発者 ID を取得
- 開発者 ID を `crm_config.yaml` に設定

2. 環境

フロント UI の `crm_config.yaml` ファイル

```
paypal:
  client_id: 'YOUR_PAYPAL_CLIENT_ID'
  client_secret: 'YOUR_PAYPAL_CLIENT_SECRET'
  mode: 'sandbox' # 'sandbox' または 'live'
```

API キーを `crm_config.yaml` に PayPal から取得した API キー

3. 設定

UI `.env` 設定

```
VITE_PAYMENT_VENDORS="stripe, paypal"
VITE_PAYPAL_CLIENT_ID=YOUR_PAYPAL_CLIENT_ID
```

`VITE_PAYPAL_CLIENT_ID` は `crm_config.yaml` の `paypal.client_id` 設定

設定表

| | 開発 | 本番 |
|--------------------|---------------------------------------|----------------------------------|
| Stripe API | <code>sk_test_ / pk_test_</code> | <code>sk_live_ / pk_live_</code> |
| PayPal API | <code>api-m.sandbox.paypal.com</code> | <code>api-m.paypal.com</code> |
| PayPal mode | <code>sandbox</code> | <code>live</code> |
| 備考 | 備考 | 備考 |

開発環境では `crm_config.yaml` の `paypal.client_id` 設定を UI `.env` の `VITE_PAYPAL_CLIENT_ID` / `VITE_STRIPE_CLIENT_ID` 設定で UI 側 API

PCI 対応

OmniCRM 対応 PCI 対応

- `iframe` 対応 Stripe Elements / PayPal Card Fields
- OmniCRM 対応 PCI 対応
- 対応 PCI 対応
- 対応 PCI 対応

□□□□□□

OmniCRM □□□□□□□□□□□□□□□□ Stripe □□□□□□□□□□□□ API □□□□□□□□□□□□
□□□□□□□ □□□□□□

□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□



□□ - □□ Square □□

```
{
  "vendor": "square",
  "vendor_payment_method_id": "sq_card_abc123",
  "payment_type": "card"
}
```

□□□□□□□□□□□□□□□□

□□□□

□□□□

PaymentMethod

□□□□□□□□□□□□□□□□

```

{
  "payment_method_id": 789,
  "customer_id": 123,
  "vendor": "stripe", // 'stripe', 'paypal'
  "vendor_payment_method_id": "pm_xxx", // ID
  "payment_type": "card", // 'card', 'paypal',
  'ach',
  "is_default": true,
  "card_brand": "visa",
  "card_last4": "4242",
  "card_exp_month": 12,
  "card_exp_year": 2025,
  "card_nickname": "My Visa Card",
  "status": "active"
}

```

PaymentAuthorization

```

{
  "authorization_id": 301,
  "customer_id": 123,
  "payment_method_id": 789,
  "vendor": "stripe", //
  "vendor_authorization_id": "auth_xxx", // 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


```

{
  "capture_id": 103,
  "authorization_id": 301,
  "customer_id": 123,
  "payment_method_id": 789,
  "vendor": "stripe",
  "vendor_transaction_id": "ch_xxx",      // 商家交易 ID
  "amount": 200.00,
  "currency": "USD",
  "status": "succeeded",                 // 'succeeded', 'failed',
  'refunded'
  "captured_at": "2025-12-27T10:30:00Z",
  "vendor_response": {},                 // 商家响应
  "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

钱包流水记录

```
{
  "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",  // 消費
  "reference_id": 103,
  "meta": {},
  "created_at": "2025-12-27T10:35:00Z"
}
```

消費

消費金額 100.00

消費

消費金額 100.00 消費 ID 103 invoice_id 103

消費金額 100.00

消費

```
{
  "invoice_id": 12345,
  "invoice_number": "INV-2025-000001",    // 发票号 INV-YYYY-NNNNNN
  "customer_id": 123,
  "title": "商品销售",
  "paid": true,                          // 是否已付
  "void": false,                        // 是否作废
  "payment_reference": "ch_xxxxx",      // 支付/退款 ID
  "payment_type": "stripe_capture",     // 支付方式
  "payment_time": "2025-12-27T10:30:00",
  "start_date": "2025-12-01",
  "end_date": "2025-12-31",
  "due_date": "2026-01-15",
  "retail_cost": 500.00,                // 零售成本
  "wholesale_cost": 250.00
}
```

发票编号: INV-YYYY-NNNNNN 商品名称: 商品名称 1 1 数量

商品名称

商品名称: 商品名称 retail_cost: 零售成本

数量

- 商品名称: "商品"
- 商品名称: 商品名称
- 商品名称: 商品名称
- 商品名称: 商品名称

商品名称

- 商品名称: 商品名称
- 商品名称: 商品名称
- 商品名称: 商品名称 - 商品名称

商品名称: 商品名称

```
// 1. 購買商品
{
  "transaction_id": 7001,
  "invoice_id": 12345,
  "retail_cost": 100.00, // 買 = 賣
  "title": "商品"
}

// 2. 退款
{
  "transaction_id": 7002,
  "invoice_id": 12345,
  "retail_cost": -100.00, // 買 = 賣
  "title": "商品 (12345)",
  "payment_type": "stripe_capture",
  "payment_reference": "ch_xxxxx"
}

// 商品$0 買 → 賣
```

payment_reference payment_type
payment_time

- invoice_id 商品 Hayden
- 商品/
- 商品

1. 商品
2. 商品

3. 準備環境変数

4. 準備環境変数

- 準備環境変数
- 準備環境

StripeのAPIを呼び出すには、StripeのAPIキーを準備する必要があります。

Stripe API

StripeのAPI URLは `/api/payments/`

StripeのAPIキーは `sk_test_1234567890`

StripeのAPI

StripeのAPI

StripeのAPI `POST /api/payments/methods`

Stripe

```
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"
}'
```

Stripe

```

{
  "customer_id": 123,
  "vendor": "stripe", // 'stripe', 'paypal'  선택
  "payment_token": "pm_xxxxx", //  SDK 
  "is_default": false, // 
  "card_nickname": "My Visa Card" // 
}

```

201

```

{
  "success": true,
  "message": " ",
  "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"
  }
}

```

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": "成功しました",
  "data": {
    "payment_method_id": 790,
    "is_default": true
  }
}
```

成功しました

成功しました

DELETE /api/payments/methods/{payment_method_id}?customer_id={id}

```
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": "成功しました"
}
```

API

1.

👉👉👉 POST /api/payments/charge

👉👉👉

```
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

準備作業は以下の通りです。

1. `POST /api/payments/invoice`

2.

1. 準備作業
2. 準備作業
3. 準備作業
4. 準備作業
5. 準備作業

3.

```
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. □□□□□□□□

[illegible]

📖 **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": "Success",
  "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,           // 
    "card_amount": 50.00,           // 
    "message": "Success $50 $200 "
  }
}
```

2

POST /api/payments/capture/{authorization_id}

1. -
- 2.
- 3.
4. /

```
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测试",
    "wholesale_cost": 100.00,
    "contract_days": 3,
    "send_email": true
  }
}'
```

测试

```
{
  "amount": 200.00,           // 金额 - 测试金额
  "metadata": {
    "invoice": true,          // 是否开票
    "create_transaction": true, // 是否创建交易
    "title": "测试",
    "description": "3测试",
    "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": { // 交易 invoice=true あり
      "transaction_id": 7001
    },
    "invoice": { // 取引 invoice=true あり
      "invoice_id": 12345,
      "invoice_number": "INV-2025-000001"
    }
  }
}

```

3 成功確認

POST /api/payments/release/{authorization_id}

成功確認

成功

```

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

POST /api/payments/topup

決済 → 決済 → 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

```
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 // 未保存 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。

curl

```
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"
  }
}'
```

レスポンスは、200 OK として返されます。

```
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"
  }
}'
```

レスポンスは `200 OK` として返されます。

```
{
  "success": true,
  "message": "成功",
  "data": {
    "transaction_id": "ch_xxxxx",
    "amount": 75.00,
    "payment_method_saved": false // 是否保存
  }
}
```

6. 退款

向 `POST /api/payments/refund`

发送

```
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"
  }'
```

返回

```
{
  "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  
'
```

请求

```
{  
  "customer_id": 123,  
  "amount": 100.00,  
  "currency": "USD", // 币种USD  
  "payment_method_id": 789 // 支付方式 - 000000000000  
}
```

返回200 OK

```
{  
  "success": true,  
  "payment": {  
    "transaction_id": "ch_xxxxx",  
    "amount": 100.00  
  },  
  "wallet": {  
    "ledger_id": 503,  
    "balance_after": 250.50,  
    "customer_transaction_id": 7001  
  },  
  "message": "成功 100.00 元"  
}
```

返回数据
customer_transaction payment_type=wallet_topup_card
retail_cost wallet_ledger_id
card_capture_topup

DELETE /crm/transaction/transaction_id/{transaction_id}
 成功すると 200 OK

成功すると

成功すると

GET /api/wallet/transactions?customer_id={id}&limit={n}&offset={n}&type={type}

成功すると

```
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"
```

成功すると

- customer_id: ID
- limit: 50
- offset: 0
- type: 'credit' | 'debit' | 'refund' | 'adjustment'

成功すると 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"
    }
  ]
}

```

□□□□

□□□□□□□□□□□□□□ delete_customer_transaction□

□□□ DELETE /crm/transaction/transaction_id/{transaction_id}

□□□

```
curl -X DELETE "https://your-  
domain.com/api/transaction/transaction_id/7001" \  
-H "Authorization: Bearer YOUR_API_KEY"
```

200 OK

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

200 OK

```
{  
  "result": "  
  "wallet_reversal": {  
    "wallet_reversed": true,  
    "wallet_amount": 50.00,  
    "wallet_ledger_id": 503  
  }  
}
```

200 OK

payment_type=wallet_topup_card

```

{
  "result": "成功",
  "wallet_reversal": {
    "wallet_reversed": true,
    "wallet_amount": 100.00,
    "wallet_ledger_id": 503,
    "refund": {
      "capture_id": 88,
      "vendor": "stripe",
      "vendor_transaction_id": "ch_xxxxx",
      "refund_id": "re_xxxxx",
      "status": "succeeded",
      "amount": 100.00
    }
  }
}

```

API/フロー

| API | wallet_ledger_id | API
ID
番号 | API名 |
|---------------------------|------------------|-----------------|-------------------------|
| API/ID | ID | ID | ID |
| API/POST
/wallet/topup | ID | API
ID | ID
wallet_topup_card |
| API/ID | ID | API
ID | ID |
| API/ID | ID | API
ID | ID |
| API/ID | API/CT ID | ID* | ID |

*API/ID → ID → API/ID card_capture_topup + service_charge API/ID
API/ID

500

```
{
  "result": "",
  "Reason": "0.0020.00"
}
```

- " "
-
-

void=true /

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} | |

결제 API

| 메소드 | API URL | 설명 |
|------|------------------------------|----------|
| POST | /api/payments/charge | 결제하기 |
| POST | /api/payments/invoice | 결제영수증 발급 |
| POST | /api/payments/topup | 충당 |
| POST | /api/payments/authorize/hold | 결제권 승인 |
| POST | /api/payments/capture/{id} | 결제권 확정 |
| POST | /api/payments/release/{id} | 결제권 취소 |
| POST | /api/payments/rental | 결제권 임대 |
| POST | /api/payments/refund | 결제권 환불 |

지갑 API

| 메소드 | API URL | 설명 |
|------|---|---|
| GET | /api/wallet/balance?
customer_id={id} | 잔액 조회 |
| GET | /api/wallet/info?customer_id={id} | 지갑 정보 + 카드 |
| POST | /api/wallet/topup | 지갑 충전
wallet_topup_card {card_no} {exp_date} |
| GET | /api/wallet/transactions?
customer_id={id} | 지갑 거래 내역 |

API

| Method | URL | Description |
|--------|--------------------------------------|-------------|
| PUT | /api/transaction/ | Transaction |
| GET | /api/transaction/ | Transaction |
| DELETE | /api/transaction/transaction_id/{id} | Transaction |

PayPal

| Method | URL | Description |
|--------|---|-------------|
| POST | /api/payments/paypal/vault/setup-token | SDK PayPal |
| POST | /api/payments/paypal/vault/finalize | PayPal |
| POST | /api/payments/paypal/vault/update-setup-token | 3DS/SCA |

API

1

30

```
# 创建支付单
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
  }
}
```

第 2 步: 创建订单

创建订单的接口地址为: `/api/orders`

```
# 請求を送信
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"
    }
  }'
```

ステップ

1. 請求額 \$149.99
2. 請求を送信
3. 請求額 → 請求
4. 請求額 → 請求額

ステップ 3 請求額

請求額 WiFi 請求額


```
# 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
- 充值备注

每 4 个充值 + 充值

充值金额充值次数充值有效期充值备注

```
# 1 결제 - 결제
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"
  }
}'

# {"authorization_id": 301, "status": "authorized"}

# 2a 결제완료 - 결제
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": "결제 - 3",
    "description": "302 결제 12월 27-30일",
    "send_email": true
  }
}'

# 2b 결제완료 - 결제
curl -X POST https://your-domain.com/api/payments/release/301 \
-H "Authorization: Bearer YOUR_API_KEY"
```

5 Stripe

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"
}
```

成功

成功

成功 400 元

```
{
  "error": "余额不足 50.00 元"
}
```

成功 400 元

```
{
  "error": "余额不足"
}
```

成功 400 元

```
{
  "error": "customer_id 或 amount 错误"
}
```

🔴 404 Not Found

```
{
  "error": "🔴 999 🚫🚫🚫"
}
```

🔴 500 Internal Server Error

```
{
  "error": "🚫🚫🚫🚫🚫🚫🚫🚫"
}
```

🔴 500 Internal Server Error

🔴 500

🔴 500 Internal Server Error

1. 🔴 Stripe/PayPal

🔴 Stripe/PayPal

🔴 Stripe/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"
}'
```

👉

- Stripe/PayPal 退款
- 退款 5-10 分钟/Stripe/PayPal 退款
- 退款
- `PaymentCapture` 退款

👉

- 退款
- 退款
- 退款
- 退款

2. 退款

退款

退款

退款

👉

- 退款
- 退款
- 退款
- `WalletLedger` 退款

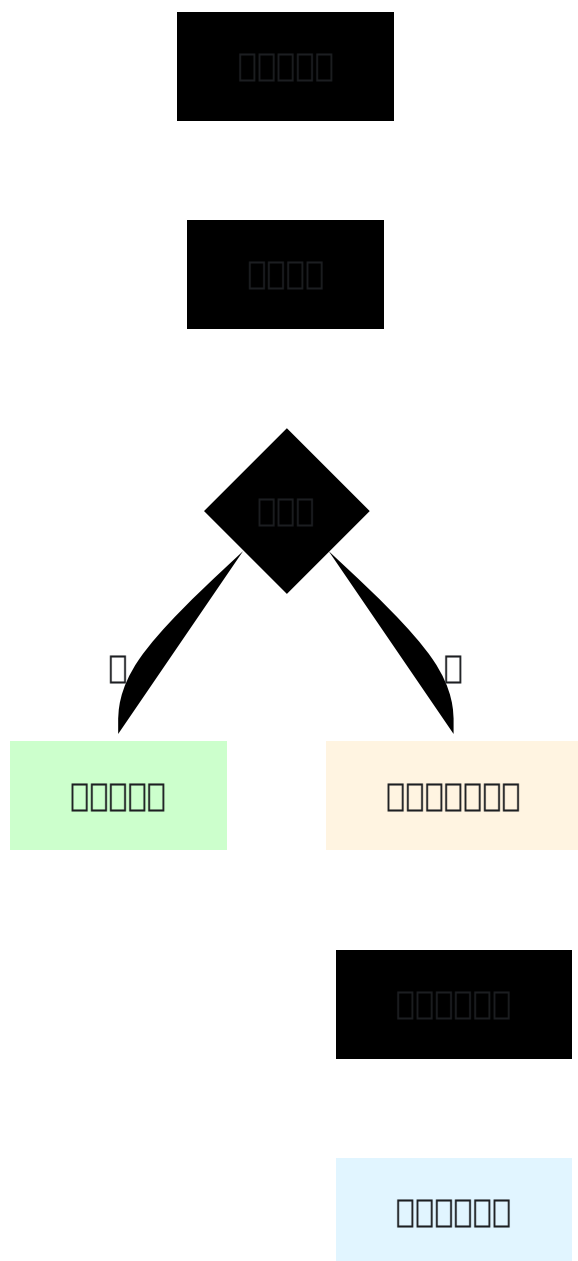
□□□□□

- 
- 
- 
- 

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

[illegible]

□□□□□□



開始

1. 開始 \$100 準備
2. 準備
3. 準備
4. 開始 \$100 準備5-10 回 + 準備
- 開始 \$100 準備
- 準備
- 準備
- 準備

□□□□□□□□□□□□□□□□

□□ □□□□□□□□□□□□□□□□□□□□

□□ **1**□**\$1** □□□□ + **\$10** □□

□□□□□□□□□ \$1□□□□□□ \$10 □□□□□□

□□□□□□□□□□□□□□□□□□

□□□□□□□□□□
\$1 □□

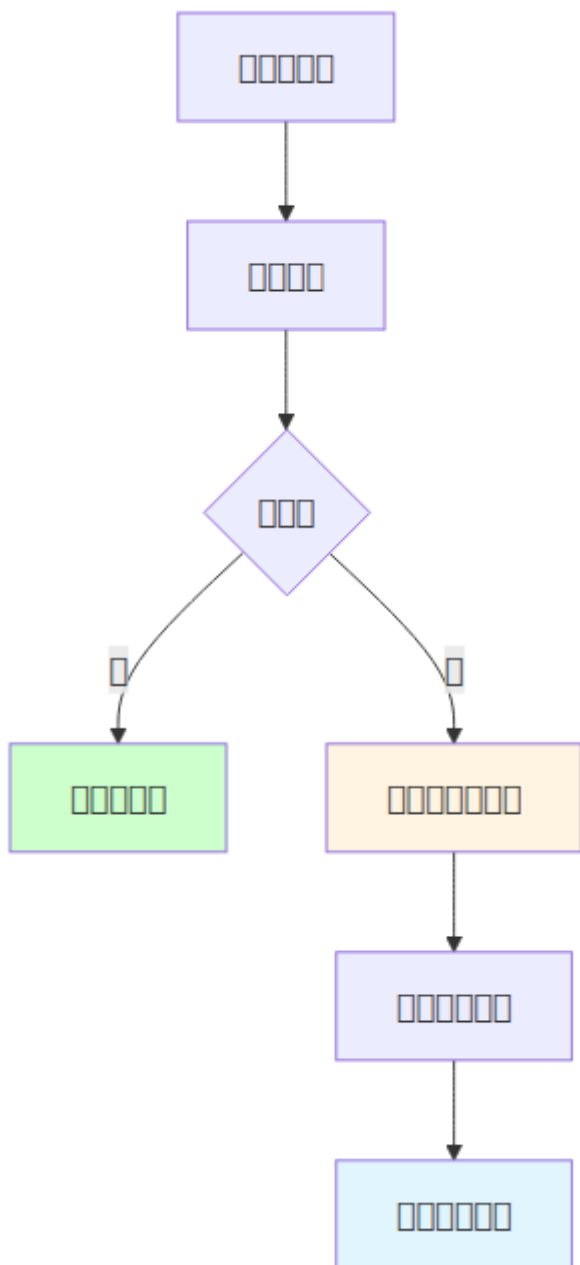
□□□

□□□□□□□□□□

□□□□ \$10

□□□\$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 関数

```
{
  "success": true,
  "message": "成功しました",
  "data": {
    "customer_id": 123,
    "service_amount": 10.00,
    "routing_mode": "hybrid",
    "initial_balance": 1.00,
    "wallet_portion_used": 1.00, // 残高 $1
    "card_portion_used": 9.00,  // カード $9 使用
    "charged_amount": 9.00,     // ← カード
    "wallet_credited": 9.00,    // ← 残高
    "wallet_debited": 10.00,    // ← カード
    "final_balance": 0.00
  }
}
```

カード 2 × \$50 残高 + \$30 カード

残高 \$50 カード \$30

残高 \$50

顧客\$50

店員
 $\$30 - \$50 = \$0$

店員

顧客

顧客

店員-\$30

顧客\$20
店員\$0
顧客\$30

API

```

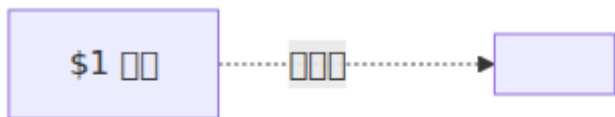
{
  "success": true,
  "data": {
    "service_amount": 30.00,
    "initial_balance": 50.00,
    "charged_amount": 0, // ← 0000
    "wallet_debited": 30.00,
    "final_balance": 20.00,
    "payment_method_used": false // ← 0000
  }
}

```

3 000000000000

00000000 \$500 000000000 \$1500

1 000000 POST /api/payments/authorize/hold00



API 000

```

{
  "authorization_id": 301,
  "amount": 500.00,
  "status": "authorized",
  "wallet_balance": 150.00,
  "wallet_to_use": 150.00, // 00000000
  "card_amount": 350.00, // 00000000
  "message": "0000 $3500000000000000000 $500 000000"
}

```

2 000000 POST /api/payments/capture/30100

0000

0000\$350

0000+\$350
000\$500

0000-\$500
0000

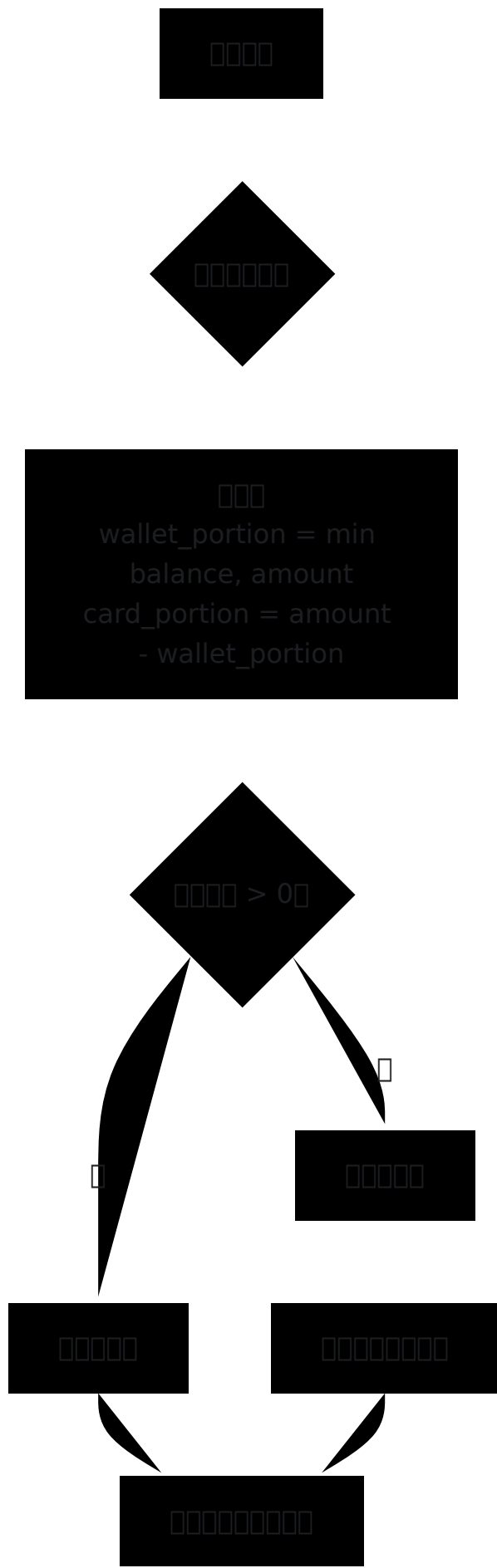
00000\$0
00000\$350

00000000

- 0 00000 \$500 0 \$350 00“00”0000 \$500
- 0 000000000000
- 0 00000000
- 0 0000000

0000

000000000000



00

00000

1. 0000000000
2. 00000 amount - wallet_balance
3. 0000 > 0000000000
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

タスク

1. 環境構築
2. 認証情報の取得
3. データの取得
4. データの保存
5. データの更新
6. データの削除
7. テストの実行
8. 完了

Playbook の実行

Ansible playbooks の実行

1. 環境構築/認証 - try/catch でエラーハンドリング
2. **authorization_id** - 認証情報の取得
3. **API** - データの取得
4. データの保存 - 2 回
5. テスト - テストの実行

Playbook の実行

タスク

API の実行 API の実行 **Authorization** タスク

Authorization: Bearer YOUR_API_KEY

API の実行

- 0000000 → 000 0000
 - 000000 → 000 Playbook 00
-



API API API API API

API

API

API

Mailjet PDF

如何配置

OmniCRM 使用 Jinja2 渲染 HTML 模板来生成各种类型的文档

模板目录

模板目录为 `OmniCRM-API/invoice_templates/`

包含以下模板

- `sample_invoice_template.html` - 普通模板
- `cifi_invoice_template.html` - 特殊模板

配置

模板配置在 `OmniCRM-API/crm_config.yaml` 文件中

```
invoice:
  template_filename: 'sample_invoice_template.html'
```

渲染

OmniCRM 使用 Jinja2 渲染

□□□□□

- `{{ 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

模板如下：

```
<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：

1. 模板文件

```
cd OmniCRM-API/invoice_templates/
cp sample_invoice_template.html your_company_invoice_template.html
```

2. **HTML/CSS**

`your_company_invoice_template.html` 模板文件

模板文件

1. 模板文件

```
<!-- 公司 Logo URL -->
![公司 Logo](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;
  }

  /* 主要按钮 */
  .btn-primary {
    background-color: #your-brand-color;
  }
</style>
```

3. 页脚

```
<footer>
  <p>公司地址</p>
  <p>123 公司大道</p>
  <p>电话 +1-555-123-4567 | 邮箱 billing@yourcompany.com</p>
  <p>ABN/12345678900</p>
</footer>
```

4. 其他


```

<div class="payment-info">
  <h3>결제정보</h3>
  <p><strong>결제URL</strong> <a href="https://yourcompany.com/pay">https://yourcompany.com/pay</a></p>
  <p><strong>계좌정보</strong></p>
  <ul>
    <li>계좌번호: 000000000000</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>결제기간: 30일</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 生成

4. 静的コンテンツの管理

静的コンテンツ

静的コンテンツ

Jinja2 テンプレート/静的コンテンツ

```
{% if total_amount > 1000 %}
    <div class="high-value-notice">
        <p><strong>高価な商品</strong> 送料 - 無料</p>
    </div>
{% endif %}

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

静的コンテンツ

静的コンテンツ

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

静的コンテンツ

静的コンテンツ

```

<!-- 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，并指定了 `tax_percentage` 为 0%。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并指定了 `tax_percentage` 为 0%。

在 `total_tax` 模板变量中，我们使用了 `tax_amount` 模板变量，并指定了 `tax_percentage` 为 0%。

```

<div class="qr-payment">
    ![[二维码]](https://api.qrserver.com/v1/create-qr-code/?
    size=150x150&data={{ payment_url }})
    <p>扫描二维码支付</p>
</div>

```

PDF 模板

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

在 CSS 中，

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

在 HTML 中，

- JavaScript
- CSS 排版排版排版
- 排版
- 排版 CSS 排版

排版排版

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

排版排版

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

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. □□□□□□
 - □□□□□□
 - □□□□□□□□

- 設定ファイル
- 設定ファイルの読み込み

3. PDF 生成

- 生成ファイル<5MB
- 生成
- 生成ファイル
- 生成ファイル

4. 送信

- 送信
- 送信
- 送信

5. 送信完了

- PDF 生成
- 送信 Mailjet 送信15MB
- GmailOutlookApple Mail 送信

送信完了

送信 API 送信

```
curl -X GET "http://localhost:5000/crm/invoice/{invoice_id}/pdf" \
-H "Authorization: Bearer YOUR_TOKEN" \
--output test_invoice.pdf
```

送信完了

送信完了

- 送信 送信
- 送信 送信

PDF 送信

- 送信 送信 CSS 送信
- 送信 送信 CSS 2.1 送信 WeasyPrint 送信 CSS

□□□□□□

- 000 00 URL 0000000000
- 000 0000 HTTPS URL0000000000

□ □ □ □ □ □ □

- 3x3 表格
- 3x3 表格 `table-layout: fixed`

□ □ □ ◆ ? ◆ ? ◆ ? □ □

- 1000 0000000000
- 1000 0000000000 Arial Times New Roman 0000000000

PDF 00000

- 簡單 HTML 簡單 WeasyPrint 簡單
- 簡單 簡單 HTML 簡單 WeasyPrint 簡單

PDF

[PDF](#) [OmniCRM PDF](#) [PDF](#) [PDF](#)

PDF 00000000

1. Invoice - PDF Cache
 - Invoice PDF
 - PDF Base64
 - PDF SHA256
 - Invoice_PDF_Cache
 - ID
 - PDF Base64
 -
 -
 -

- 2. 詳細 - 詳細検索機能
 - 詳細 invoice_id 詳細 PDF
 - 詳細検索機能詳細検索 PDF
 - 詳細 last_accessed 詳細検索機能
- 3. 詳細 - 詳細検索機能 PDF 詳細
 - 詳細検索機能/詳細検索機能
 - 詳細検索機能
 - 詳細検索機能

詳細

- 詳細 - 詳細検索機能 PDF 詳細検索機能
- 詳細 - 詳細検索機能 PDF 詳細検索機能
- 詳細検索 - 詳細 PDF 詳細 CPU 詳細
- 詳細 - 詳細検索機能詳細検索機能

詳細

詳細 PDF 詳細検索機能詳細検索機能

- 詳細検索機能 90 詳細検索機能
- 詳細検索機能 30 詳細検索機能
- 詳細検索機能詳細検索機能

API 詳細

詳細 API 詳細 UI 詳細検索機能

- 詳細検索機能 PDF 詳細検索機能
- 詳細検索機能
- 詳細検索/詳細検索機能

詳細 詳細検索機能詳細検索機能

```
-- 詳細検索機能 PDF MySQL 詳細  
DELETE FROM Invoice_PDF_Cache;
```


📄 `crm_config.yaml` 📄📄📄📄📄📄📄📄

📄📄📄📄

📄📄📄📄📄📄📄📄📄📄

🔗🔗📄📄📄

1. 📄📄 📄 → **[📄📄📄]**
2. 📄 📄 📄
3. 📄📄📄📄📄📄📄

📄📄📄📄📄

1. 📄📄 📄 → 📄📄📄📄📄
2. 📄📄📄📄📄📄📄

📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

- 📄📄 - 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

- 000000 - 00000000000000000000
- 000000 - 0000000000000000
- 000000 - 0000000000000000

□□□□□□

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

□□□□□□

- " " " " " " " " " " " " " "
- " " " " " " " " " " " " " "
- " " " " " " " " " " " " " "
- " " " " " " " " " "

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- **ID** - ID
- - /
- - "N/A"
- -
- -
- -
- -
- -

- **↓ ()** - PDF
- **()** -
- **()** -
- **ⓧ ()** -
- **()** - Stripe Stripe

"+"

- 00000 - 00000000000000000000000000000000
- 00 - 0000/0000000000 "00 [00]"0
- 00000 - 000000000000 14 000
- 00000 - 0000000000000000
- 000 - 0000000000000000
- 00000 - 0000000000000000000000000/00000000

□□□□□

- ✓ (00000) - 0000000000000000
- ✗ (0000) - 0000000000000000
- 00 - 000000000000
- 0000 - 00000000
- 0000000000000000000000000000000000
- 00/0000000000000000000000000000000000

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

1. 00000000000000000000
2. 000000000000/000000
3. 00000000000000000000
4. 000000000000000000
5. 0000 PDF 000
6. 000000000000 invoice_id 00000
7. 000000000000000000000000
8. 0000000 "000" 0000

□□□□□

[illegible][illegible][illegible]

- PDF - 文件格式

PDF

PDF 文件 (↓) 文件名称 "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 (↓) 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. Mailjet の `api_crmCommunicationCustomerInvoice` エンドポイントにリクエストを送信する

4. 請求書データを取得する

- 顧客の PDF 請求書
- 請求書の ID
- 請求書の発行日
- 請求書の支払期日
- 請求書の総額
- 請求書の PDF へのリンク

5. 請求書の URL を取得する

請求書の URL

請求書の URL は "URL" フィールドに返されます

請求書の URL

- `{{ var:customer_name }}` - 顧客の名前
- `{{ var:invoice_number }}` - 請求書の ID
- `{{ var:invoice_date }}` - 請求書の発行日
- `{{ var:due_date }}` - 請求書の支払期日
- `{{ 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 }}` - 請求書の支払 URL

請求書の URL

- 請求書の URL - `crm_config.yaml` の Mailjet API キー
- 請求書の URL - 請求書の発行日
- PDF** 請求書 - 請求書の PDF を生成する

請求書の URL

請求書の URL (URL) を取得する

請求書の URL

🔒🔒🔒🔒

🔒 🔒🔒🔒 (🔒) 🔒🔒🔒🔒🔒🔒

🔒🔒

- 🔒🔒🔒🔒 🔒🔒
- 🔒🔒🔒🔒 **Stripe** 🔒
- 🔒🔒🔒🔒🔒 `payment_reference` 🔒Stripe 🔒🔒🔒 ID🔒
- 🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒🔒

🔒🔒🔒🔒

1. 🔒🔒🔒🔒 Stripe 🔒
2. 🔒🔒🔒🔒 (🔒)
3. 🔒🔒🔒🔒🔒🔒🔒

4. 🔒 "🔒🔒🔒"

5. 🔒🔒 Stripe 🔒🔒

- 🔒 Stripe API 🔒🔒
- 🔒 Stripe 🔒🔒🔒🔒
- 🔒🔒🔒🔒 `refund_reference`

6. 🔒🔒🔒🔒🔒🔒🔒🔒 "🔒🔒"

7. 🔒🔒🔒🔒

🔒🔒🔒🔒🔒

- 0000000000000000
- 0000 "0000"
- 0000000000
- 00000000000000003-7 000000
- Stripe 0000000000

000000

- 0000000000POS 00000000000000000000
- 000000000000000000
- 00000000

0000000000

00

00000000000000000000

- 00 ID
- 00000
- 0000000000000000

00

00000000000000

000000

- 00000 - 000000000000
- 00000 - 000000000000

000000

- 00000 - 000000000000
- 00000 - 0000000000000000

- 000000000000

8. 00 "0000"000000000000"00000150"

9. 0000000000000000

10. 0000000000000000

11. 0000000000 "000000" 0000000000000000

12. 0000000 PDF 0000000000000000

0000 200000000000

1. 00000000000 + 00000000000

2. 0000000000000000

3. 0000000000000000

- 00 "+ 0000000"
- 000"00000 - 2025 0 1 0"
- 00/0001 0 1 00 31 0
- 00002 0 15 0
- 0000000000 0000000000
 - 0000000000 X 00
 - 0000000000 X 00
 - 00000000000000
- 00000000000000
- 00 "0000"000000000000

4. 000000000000000000

- 0000 "+ 0000000"
- 000"00000 - 2025 0 1 0"
- 00/0001 0 1 00 31 00000000
- 000000000
 - 0000000000000000
 - 00 X 0000000000
 - 00000000000000
- 00 "0000"

5. 000000000000000000

- 0000 "+ 0000000"
- 000"00000 - 2025 0 1 0"

- □□□□□□□□□□□□□□□□
- □□ "□□" □□□□□□
- □□ "□□□□"

6. □□□□□□□□□□

□□□□ **3** □□□□□□□□□□

- [illegible]

[illegible]

1. 
2. 
3. 
 - 
 - 
 - 
 - 

4. 註冊帳號
5. 登入系統並進行驗證
6. 將 "帳號" 與密碼進行驗證
7. 將  與密碼進行驗證/密碼進行驗證
8. 登入系統並進行驗證

步驟 5 登入系統

1. 登入系統
2. 登入系統
3. 登入系統 登入 登入
4. 將 "+" 與密碼進行驗證
5. 登入系統
 - 登入系統
 - 登入系統
 - 登入系統 + 7 登入
 - 登入 "登入"
6. 登入系統
7. 登入系統
8. 將 **X** 與密碼進行驗證
9. 將 "登入"
10. 登入系統
11. 登入 Stripe 登入

步驟 6 登入系統

1. 登入系統
2. 登入系統
3. 登入系統 登入系統
4. 登入系統
 - 將 ID 與密碼進行驗證
 - 登入系統
 - 登入系統
 - 登入系統

5. 0000000000000000
6. 000000 PDF00000000 "00 **PDF**" 00
7. 000000000000000000 "000000" 00
8. 0000000000

0000 **7**0000000000

1. 0000000000
2. 000000000000
3. 0000000000000000
4. 00💎💎000000
5. 00000000000000
6. 00000000 "00" 00
7. 0000
8. 0000000000
9. 000000000000000000
10. 0000000000000000
 - 00000000000000000000
 - 00000000
11. 000000000000000000

0000 **8**0000000000

1. 000000000000
2. 00000 00 00
3. 0000000000
4. 00000000000000
5. 000000000
6. 00000000 "0000" 00
7. 00 "00" 0000
8. 000000“0000 2025-01-15”
9. 00 "0000"
10. 000000000000 "0000"
11. 000000000000

12. 將以下字串 "1234567890"

第 9 章

1. 將以下字串 "1234567890"
2. 將以下字串 "1234567890" 與 Stripe 字串
3. 將以下字串 "1234567890"
4. 將以下字串 "1234567890"
5. 將以下字串 "1234567890"
6. 將以下字串 "1234567890" 與 Stripe 字串
7. 將以下字串 "1234567890"
8. 將以下字串 "1234567890" 與 Stripe 字串
9. 將以下字串 "1234567890" 與 Stripe 字串
10. 將以下字串 "1234567890" 與 Stripe 字串
11. 將以下字串 "1234567890" 與 Stripe 字串

第 10 章

第 10 章 - 第 10 章

- 將以下字串 "1234567890" 與 Stripe 字串
- 將以下字串 "1234567890" 與 Stripe 字串

第 11 章 PDF 字串

- 將以下字串 "1234567890" 與 Stripe 字串
- 將以下字串 "1234567890" 與 Stripe 字串

第 12 章 Stripe 字串

- 將以下字串 "1234567890" 與 Stripe 字串
- 將以下字串 "1234567890" 與 Stripe 字串

第 13 章

- 將以下字串 "1234567890" 與 Stripe 字串

- 設定 環境変数/Stripe 環境変数

環境変数

- 設定 Mailjet API 環境変数
- 設定 設定 `crm_config.yaml` 設定 Mailjet 環境変数

環境変数

- 設定 環境変数/POS/Stripe 環境変数
- 設定 環境変数 Stripe 環境変数

PDF 環境変数

- 設定 PDF 環境変数
- 設定 環境変数 PDF 設定 `DELETE FROM Invoice_PDF_Cache WHERE invoice_id = X;`

環境変数

- 設定 環境変数
- 設定 環境変数 環境変数

環境変数

- 設定 環境変数
- 設定 環境変数 環境変数

環境変数

- 設定 環境変数
- 設定 環境変数 環境変数

環境変数

- 設定 環境変数
- 設定 環境変数 環境変数

環境変数

- [API 使用ガイド](#) - API の使用方法について詳しく説明しています。
- [API 仕様書](#) - API の仕様について詳しく説明しています。

お問い合わせ

- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。
- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。

お問い合わせ

- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。
- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。

お問い合わせ

- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。
- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。

お問い合わせ

- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。
- [お問い合わせ](#) - お問い合わせの受付時間やお問い合わせのやり方について詳しく説明しています。

お問い合わせ

- [integrations_mailjet](#) - Mailjet との連携について詳しく説明しています。
- [administration_configuration](#) - 管理画面の構成について詳しく説明しています。
- [payments_transaction](#) - 決済処理について詳しく説明しています。
- [payments_process](#) - 決済処理について詳しく説明しています。
- [basics_payment](#) - 決済処理について詳しく説明しています。
- [payment_system_guide](#) - 決済 API の使用ガイドについて詳しく説明しています。



□ □

□□□□□□□□□□□□□□□□“□□”□□

☐ “☐☐” ☐ ☐ ☐ ☐

□ □

[illegible]

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

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

□ □

[illegible][illegible][illegible]

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □ □ □

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

□ □ □ □ □ □ □

1. 000 00 → 00000000
2. 000000000000

| | | | | | | |
|--|--|--|--|--|--|--|
| | | | | | | |
|--|--|--|--|--|--|--|

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

□□□□□□

- 0000 - 0000000000000000
- 000000 - 00000000000000
- 00000 - 000000000000
- 000000 - 000000000000

5/5

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

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

□□□□□□□□□□□□□□□□

1111

- **ID** - 0000ID
- 00 - 000000
- 00 - 00000
- 00 - 0000000
- 00 - 0000000000000000
- 00 - 00000000000000ID00000000
- 00 - 0000000000000000000000000000

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

□□□□□□□□□□ : □□□□□□□□

- 圖文 - 圖文掃描器
- 圖文**PDF** - 圖PDF掃描器
- 圖文 - 圖文掃描器

4444

□ □ □ □ □ □ □ □

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
|--|--|--|--|--|--|--|--|

□ □ □ □ □ □ □ □ □ □ □ □

- 00000000 - 0000000000000000
- 00000000 - 00000000
- 00000 - 0000000000000000
- 00000 - 0000000000000000SIM00000
- 00000000 - 0000000000000000
- 00000 - 0000000000000000

□□□□□□□/□□□□

□□□□□□□□□□□□

- □□□□ - □□□□□□□
- □□□ - □□□□□□□/□□□□□
- □□□□□□□ - □□□□□□□□□□□
- □□□□ - □□□□□□□
- □□ - □□□□□□□□□
- □□ - □□□□□□□□□

□□□□□□□

□□ "+ □□□□" □□□□□□□□□□

□□□□□□□□□□

□□□□□□□□/□□□□

5/5

- 0000 - 0000000000000000/000
- 0000 - 00000000000000000000000000000000
- 00 - 0000???
- 00 - 00000000
- 0000 - 0000000000000000
- 0000 - 0000000000000000
- 000000 - 00000000000000000000000%0
- 00 - 0000000000000000
- 00 - 0000000000000000
- 0000 - 000000000000

111

- 
- 
- 

□□□□□□

1.
2.

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

00000000

- 000000 void = true
- 0000000000000000

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

零售總額

- 零售總額 10% 零售總額 \$50.00
- 零售總額 $\$50.00 \times 0.10 = \5.00
- 零售總額 \$5.00 (10%)

零售 **NIL**/零售總額

- 零售總額 0 零售總額
- 零售總額 0%
- 零售總額 “-”

データベース設計

データベース設計

データベース設計

データベース設計

- データベース設計
- データベース設計
- データベース設計
- データベース設計
- データベース設計

データベース設計

- データベース設計
- データベース設計
- データIDデータベース設計
- データベース設計✓
- データベース設計

データベース設計

データベース設計

1. データベース設計
2. データベース設計
3. データベース設計
4. データ `invoice_id` データベース設計
5. データベース設計“データベース”

データベース `payments_invoices` データベース設計

Playbooks

Ansible playbooks OmniCRM

API

11

OmnicRM Ansible Playbooks ██████████ CRM ███ API █████

1. $\frac{m}{n}$ - $\frac{m^2 + n^2}{m^2 - n^2}$
2. $\frac{m}{n} = \frac{m^2 + n^2}{m^2 - n^2}$

[illegible]

Playbooks

```

####  ????? #### retail_cost wholesale_cost
retail_setup_cost ##### Playbook #####
####

```

Ansible playbook

- [illegible]

API 仕様書

```
# 商品 retail_cost が $99.00 から $79.00 に変更された
- name: 商品 retail_cost の変更
  set_fact:
    charge_amount: 79.00

- name: 商品を登録
  uri:
    url: "http://localhost:5000/crm/payments/authorize/hold"
    method: POST
    body_format: json
    body:
      customer_id: "{{ customer_id | int }}"
      amount: "{{ charge_amount | float }}" # 商品 retail_cost
      retail_cost
      # ...
```

API 仕様書

```
# 商品登録
- name: 商品登録
  uri:
    url: "http://localhost:5000/crm/customer/{{ customer_id }}"
    # ...
  register: customer_info

- name: 商品価格
  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: 商品最終価格
  set_fact:
    final_price: "{{ (base_price | float) * (1 - (discount_percent | float / 100)) | round(2) }}"
```

Ansible 入門

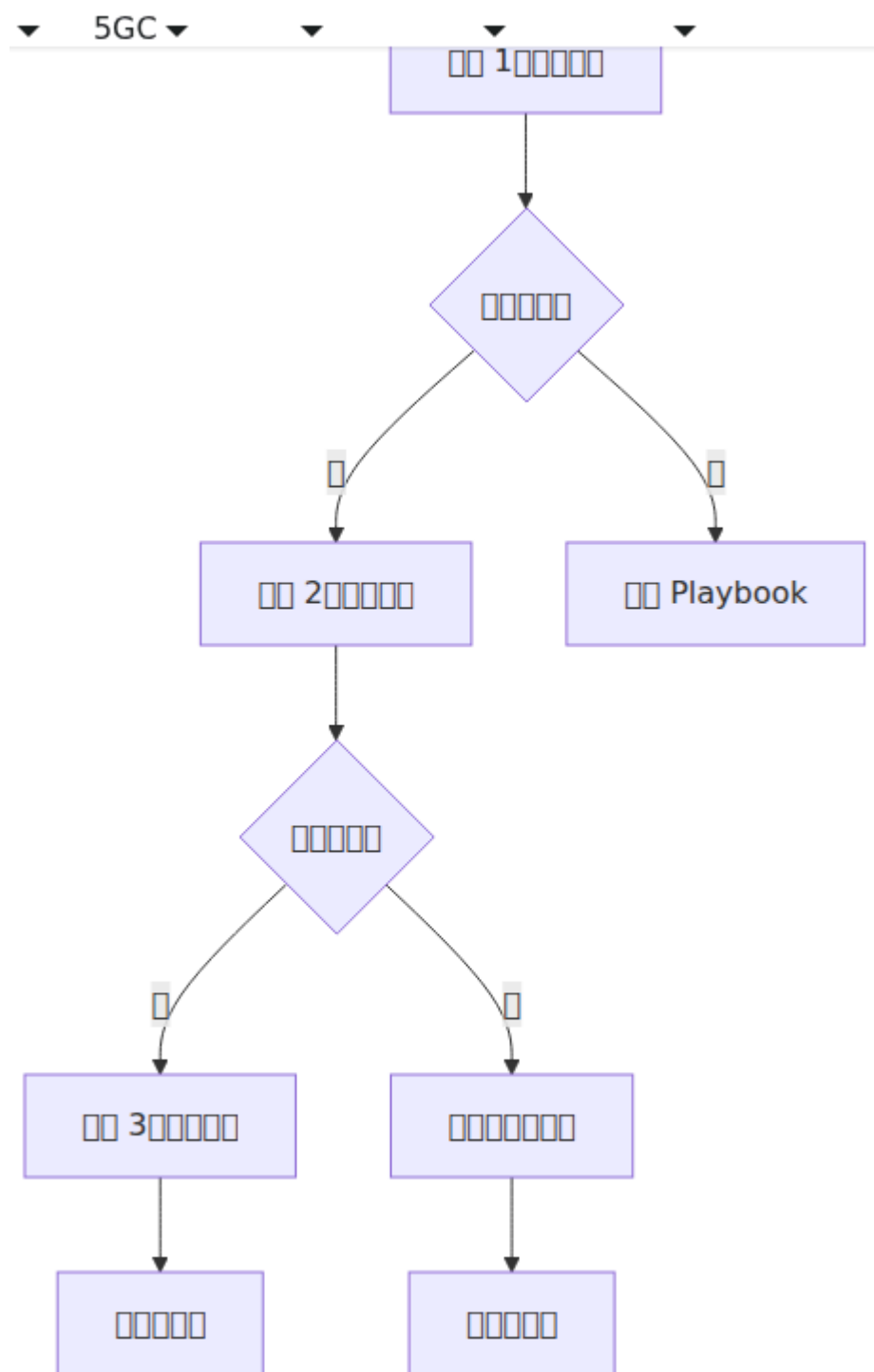
```
# 設定ファイル wholesale_cost と retail_cost
- name: 設定ファイル
  set_fact:
    charge_amount: >-
      {% if customer_info.json.is_wholesale_partner %}
      {{ api_response_product.json.wholesale_cost | float }}
      {% else %}
      {{ api_response_product.json.retail_cost | float }}
      {% endif %}
```

Ansible Playbook は、Ansible の基本となる概念で、CRM の設定/更新
などの Playbooks を実行するための Ansible の基本となる概念

Ansible のインストール

Ansible のインストールは、Ansible の基本となる概念で、CRM の設定/更新
などの Playbooks を実行するための Ansible の基本となる概念

□□□□



□□□□

□□□□□□□□□□

□□ **1**□□□ - □□□□□□□□□□

□□ **2**□□□ - □□ OCS □□/□□□□

00 **3**000 - 0000000000
00 - 000000000000

0000

00000 play_topup_charge_then_action.yaml 000000


```

- name: Play Topup - 充值 Topup
  hosts: localhost
  gather_facts: no
  become: False

  tasks:
    - name: 设置 crm_config 变量
      ansible.builtin.include_vars:
        file: "../crm_config.yaml"
        name: crm_config

    # 获取 token
    - name: 获取 CRM API 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 服务信息
      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: 注册
  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 }}"

# 200CS 配置
- block:
  - name: "200CS CGRateS 配置"
    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: 断言断言断言
  assert:
    that:
      - action_execute_response.status == 200
      - action_execute_response.json.result == "OK"

# 断言 3次断言断言 - 断言断言断言断言
- name: "断言 3次断言断言断言"
  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: 断言断言断言
  assert:
    that:
      - api_response_capture.status == 200
      - api_response_capture.json.success == true

rescue:
  # 断言断言断言断言断言断言断言断言
  - name: "断言断言断言断言"
    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: Deploy playbook
  fail:
    msg: "OCS provisioning {{ authorization_id }} failed"

```

API

\$1000000 \$1000000000

\$9000000000

1. □□/□□□□

```
POST /crm/payments/authorize/hold
```

□ □

\$150 \$500 \$350

□ □ □ □

```

{
  "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 - xxxxxx ID
- **amount**xxxx - xxxxxxxxxxxx
- **currency**xxxx - xxxxxxxxxxxxxxxx
- **payment_method_id**xxxx - xxxxxxxx
- **metadata**xxxx - xxxxxxxxxxxx
 - **description** - xxxx
 - **service_id** - xxxxxx
 - **site_id** - xxxxxxxxxxx
 - **product_id** - xxxxxx
 - **user_id** - xxxxxxxxxxxx
 - **title** - xxxx
 - **wholesale_cost** - xxxxxxxxxxxx
 - **invoice** - xxx **true**xxxxxxxxxx
 - **contract_days** - xxxx
 - **send_email** - xxx **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"
  }
}
```

参数

- authorization_id 充值授权ID
- card_amount 充值金额
- 充值币种

2. 充值

POST /crm/payments/capture/{authorization_id}

请求参数: invoice: true

返回

```
{
  "metadata": {
    "provisioning_status": "success",
    "cgr_action": "Action_Topup_Standard",
    "additional_info": ""
  }
}
```

####

```
{
  "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}`

00000000000000000000000000000000/000000000000

0000

```
{
  "metadata": {
    "release_reason": "provisioning_failed",
    "error_details": "OCS 000000"
  }
}
```

000

```
{
  "success": true,
  "message": "000000"
}
```

000000

1. 0000000000000000
2. 0000000000000000
3. 0000000000000000
4. 000000
5. 000000

000

00000000

000000000000000000000000000000000000 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
# 0000000000000000

- 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:
      - "0000: ${{ monthly_cost }}"
      - "0000: {{ days_in_month.stdout }}"
      - "0000: {{ days_remaining }}"
      - "000000: ${{ 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次回の請求日
  set_fact:
    service_start_date: "2024-01-15"
    next_billing_date: "2024-02-01"

- name: 残日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: 残日計算の結果を30に設定
  set_fact:
    billing_period_days: 30

- name: 比例計算のシェルスクリプトを実行
  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を設定
  debug:
    msg:
      - "サービス開始日: {{ service_start_date }}"
      - "次回の請求日: {{ next_billing_date }}"
      - "残日: {{ days_until_billing.stdout }}"
      - "比例計算結果: ${{ pro_rata_cost }}"

```

例1: 1ヶ月の請求

例 1: 1ヶ月の請求

- サービス開始日: 2024年1月15日
- 月額料金: \$60.00
- 1月の総日数: 31
- 請求期間: 17日間 (1月15日から1月31日まで)
- 請求金額: $\$60.00 \times 17 \div 31 = \32.90

例 2: 2ヶ月の請求

- 100000000 10 000
- 000000 \$30
- 000000 \$50
- 000000030
- 00000021
- 000000 \$20
- 000000 $\$20.00 \times 21 \div 30 = \14.00

00000

1. 000000/00000

000000000000/000000000000

```
- block:
  # 0000
  - name: 0000
    uri: ...

  # 0000000000
  - name: 0000
    uri: ...

rescue:
  # 000000000000
  - name: 000000
    uri: ...

  - name: 00 playbook
    fail:
      msg: "000000000000"
```

2. 00000 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 为响应/请求头

```

- name: 设置 authorization_id 为响应/请求头
  set_fact:
    authorization_id: "{{
api_response_authorization.json.data.authorization_id }}"

```

4. 设置元数据

设置元数据为响应/请求头

```

metadata:
  description: "测试元数据"
  service_id: "{{ service_id | int }}"
  product_id: "{{ product_id }}"
  user_id: "{{ initiating_user | int }}"
  title: "测试元数据"
  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

设置变量为响应/请求头


```
- name: 測試測試測試
  set_fact:
    final_cost: "{{ (base_cost | float * multiplier | float) |
round(2) }}"
```

6. 測試測試測試

測試測試測試測試測試測試測試

```
- name: 測試測試測試
  set_fact:
    payment_method_id: "{{ api_response_payment_methods.json |
json_query(query) }}"
  vars:
    query: "data[?is_default==`true`].payment_method_id | [0]"

- name: 測試測試測試
  assert:
    that:
      - payment_method_id is defined
      - payment_method_id != ""
      - payment_method_id != None
  fail_msg: "測試 {{ customer_id }} 測試測試"
```

測試

1. 測試測試測試

測試測試測試測試

1. 測試測試測試
2. 測試
3. 測試 OCS/CGRateS 測試
4. 測試 CRM 測試
5. 測試
6. 測試測試測試測試

```
play_topup_charge_then_action.yaml
```

□□ **2**□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

1. □□□? ? ?
2. □□□□□□
3. □□□□□□□□□□□□□□□□
4. □□□□□□□□□□

play_simple_service.yaml 202-232

3000000000

□ □ □ □ □ □ □ □ □ □ □ □

1. 000000
2. 00 CGRateS 00
3. 000000
4. 000000000000

4 ActionPlan

□ □ □ □ □ □ □ □ □

1. `http_post` `http_post`
2. `monthly` `monthly` `ActionPlan`
3. `ActionPlan` `ActionPlan`
4. `CGRateS` `CGRateS`
5. `playbook` `playbook`

□□□□

□□□□

□□□ □□□□□□□□

□□□□□

- □□□□□□□□□□
- □□□□
- □□□□□□
- □□ ID □□□

□□□□□ □□□□□□□□□□□□□□

□□□□□□□□□□□□

□□□ □□□□□□□□□□□□

□□□ □□□□□□□□□□ - □□□□□□□□□□□□□□

□□□□□

- □□□□□□□□□□ 7 □□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□

□□□□□□□

□□□ □□□□□□□□□□□□

□□□ □□□□□□□□□□ invoice: true

□□□□□

- □□□□□□□□□□ invoice: true□□□

- 0000000000000000

0000000000

000 00000000000000

00000

- 000000000000/0000/00000
- 0000000000000000
- 000000

00000

- 0000000000000000000000
- 00000000000
- 0000000000
- 00000000000

0000000000

0000

00000000000000

1. 00000000 - 00000000/PayPal

```
- name: Refund Payment
  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. Payment - Payment Gateway Integration

Payment Gateway Integration

- Payment Gateway
- Payment Gateway Integration
- Payment Gateway

Payment Gateway Integration

Payment Gateway

Payment Gateway Integration

- Stripe ☐ ACH ☐
- PayPal ☐ PayPal ☐

Payment Gateway playbooks ☐ Square ☐ Adyen ☐ Braintree ☐

Payment Gateway

-  - Payment Gateway
-  - Stripe/PayPal ☐ 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 專案

OmniCRM 專案

OmniCRM 專案是一個基於 **RBAC** 的 CRM 系統，主要功能包括：
• `view_customer`：查看客戶資訊
• `create_inventory`：創建庫存記錄

專案

RBAC 模型

- 系統使用者可以查看所有客戶資訊
- 系統使用者可以創建庫存記錄
- 系統使用者可以刪除庫存記錄
- 系統使用者可以“刪除客戶資訊”功能，但需要確認客戶是否已刪除/停用

專案

- 系統使用者可以查看 OmniCRM 專案
- 系統使用者可以查看 `view_customer` 和 `delete_product` 功能

- □□□□□□□□□□□Admin□Support□Finance□
- □□□□□□□□□□□□□□□□

JWT API IP /

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

OmniCRM

□ □ □ □

[illegible]

0000 - 0000000000000000AdminSupportFinance00000000CRM00000000
0000000000

```
root@kali:~# cat /etc/passwd | grep admin
```

OmniTouch

API 与 Web UI 的交互

API 与 Web UI 的交互

Web UI

1. API 与 Web UI 的交互
2. API 与 Web UI 的交互

3. API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
 - API - API 与 Web UI 的交互
4. API 与 Web UI 的交互
5. API 与 Web UI 的交互

API 与 Web UI 的交互

- 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 仕様書

ユーザー登録

```
{
  "username": "jane.doe",
  "email": "jane.doe@company.com",
  "password": "TempP@ssw0rd!",
  "first_name": "Jane",
  "last_name": "Doe",
  "role": "Support,Finance"
}
```

レスポンス

POST /auth/roles/{role_id}/users/{user_id}

API 仕様書

ユーザー検索

GET /auth/users

API 仕様書

検索

GET /auth/users/search?search={query}&filters={"role":
["Support"]}&page=1&per_page=50

レスポンス

- ユーザー
- ページネーション
- 検索条件
- 2FA
- 権限

API仕様

GET /auth/users/{user_id}

返されるデータ

- ユーザー名
- ユーザーID
- 2FAフラグ
- ユーザータイプ
- ユーザー権限

レスポンス例

[[{"user_id": 1, "username": "admin", "2fa": false, "user_type": "admin", "permissions": ["admin"]}, {"user_id": 2, "username": "user", "2fa": true, "user_type": "user", "permissions": ["user"]}]]

エラー

Web UI

1. ログイン → ログアウト
2. ログイン成功メッセージ
3. ユーザータイプ
◦ admin - ユーザー権限をTier2_Supportに設定
◦ user - ユーザー権限をTier1_Supportに設定
4. ログアウト成功メッセージ
5. ユーザー権限をTier2_Supportに設定

API

POST /auth/roles

admin

レスポンス

```
{
  "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": []
}
```

| | | | | | | |
|--|--|--|--|--|--|--|
| | | | | | | |
|--|--|--|--|--|--|--|

[illegible]

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. $\square\square\square\square\square\square\square * \text{"} \square\square\square \text{"} **$
4. $\square\square\square\square\square\square\square\square\square\square\square$
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. □□□□

2. **"X" " **

3. □□□□

API

```
DELETE /auth/roles/{role_id}/permissions/{permission_id}
```

111

```
DELETE /auth/roles/45/permissions/31
```

```
create_provision
```

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
|--|--|--|--|--|--|--|--|

□□□□□□□□

Web UI

1. $\square\square\square\square\square\square\square \rightarrow \square\square\square\square$
2. $\square\square\square\square\square\square\square$
3. $\square\square\square\square\square\square\square\square$
4. $\square\square^{**}\square\square^{**}$

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「パスワード」を

保存する

Web UI

1. ログイン → ログイン
2. ログイン
3. ログイン/パスワード
4. **“”**

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}

API 說明

API 說明

GET /auth/roles/{role_id}/users

API 說明

API 說明

GET /auth/users/{user_id}

API 說明 roles API 說明

API 說明

OmniCRM API 說明

API 說明

API 說明

1. API 說明 “API 說明”
2. API 說明 - API 說明
3. API 說明 - API 說明 1000
4. API 說明 - API 說明
5. API 說明 - API 說明
 - API 8000
 - API 說明
 - API 說明
 - API 說明
 - API 說明
6. API - API 說明

API 說明

1. API 說明


```
{
  "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

2FA

Web UI

1. →
- 2.
3. **"2FA"**
4.
 - TOTP
 - 10
5. Google Authenticator Authy
- 6.
- 7.
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

API仕様

- 2FAコードは10桁
- 1分間隔で有効
- 1ユーザーあたり10回まで
- 10分間隔で有効

2FA API

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

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

注意事項

- **TOTP**方式で生成された6桁のコード
- 10分間隔で有効

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
- コード**“バックアップコード”**
- バックアップコードを生成
- バックアップ

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

- バックアップコード → 2FA
- バックアップ
- コード**“2FA”**
- バックアップ
- 2FAバックアップコードを生成
- バックアップコードを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CRUD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`: 删除Grafana删除/删除
Grafana删除

删除/删除

- 删除
 - `view_customer`、`create_customer`、`update_customer`、`delete_customer`
 - 删除 `view own customer`
- 删除
 - `view_customer_site`、`create_customer_site`、`update_customer_site`、`delete_customer_site`
 - 删除 `view own customer site`
- 删除

- `view_customer_contact` `create_customer_contact`
`update_customer_contact` `delete_customer_contact`
- `view own customer contact`
- `view_customer_attribute` `create_customer_attribute`
`update_customer_attribute` `delete_customer_attribute`
- `view own customer attribute`
- `view_customer_tag` `create_customer_tag` `update_customer_tag`
`delete_customer_tag`
- `view own customer tag`
- `view_customer_service` `create_customer_service`
`update_customer_service` `delete_customer_service`
- `view own customer service`
- `view_customer_activity` `create_customer_activity`
`update_customer_activity` `delete_customer_activity`
- `view own customer activity`



- **Stripe**
 - `view_customer_stripe_card` `create_customer_stripe_card`
`update_customer_stripe_card` `delete_customer_stripe_card`
 - `view own customer stripe card`
- `view_customer_transaction` `create_customer_transaction`
`update_customer_transaction` `delete_customer_transaction`
- `view own customer transaction`
- `view_customer_invoice` `create_customer_invoice`
`update_customer_invoice` `delete_customer_invoice`
- `view own customer invoice`

□□

- *view_communication* □ *create_communication* □ *update_communication* □ *delete_communication*
- □□□□ *view own communication*

□□□□□

- □□
 - *view_inventory* □ *create_inventory* □ *update_inventory* □ *delete_inventory*
 - □□□□ *view own inventory*
- □□□□
 - *view_inventory_template* □ *create_inventory_template* □ *update_inventory_template* □ *delete_inventory_template*
 - □□□□ *view own 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 own 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

Request

```
{ "user_id": 42 }
```

Response

POST /auth/stop_impersonation

Notes

- Only users with the `admin` role can impersonate other users.
- The impersonated user's session will be terminated after 15 minutes.
- The impersonated user's session will be terminated after 15 minutes.
- The impersonated user's session will be terminated after 15 minutes.

Grafana

OmniCRM Grafana access is managed by the `grafana_access` table in the Grafana database.

Grafana

The `grafana_access` table in the Grafana database contains the following columns:

OmniCRM Grafana - Grafana

Grafana

- `id`
- `username`
- `password`
- `email`

OmniCRM

Grafana 与 OmniCRM 集成 CRM 系统

步骤

1. **Grafana** 安装 - 访问 `/grafana` 安装
2. 创建 **ID** - Grafana URL 包含 `/grafana/d/abc123/dashboard-name` ID `/d/` 包含 `abc123`
3. 创建 - 创建 ID 包含
4. 创建 - 创建 OmniCRM “” 包含 OmniCRM

步骤

步骤

- **ID** - Grafana URL
- - OmniCRM

步骤

Grafana

grafana_access Grafana

-
-
-
-

步骤

步骤

□□□□□□□□ □□□□□□□□□□□□□□□□

“□□□□...”□□□□□□□□□□ □□□□□□/□□□□□□□□□□□□□□□□

□□□□□□□**Grafana**□□□□ □□□□□□□ grafana_access□□□□□□ /grafana□□□□□□
Grafana□□□

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

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

□□□□□□

□□□□□□

- 
- 
- 
- 

| | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

- 100ID
- 10000000
- 100000000/10000
- 10000000
- 1000000

□ □ □ □ □ □ □

1. □□□ □ → □□□□
2. □□□□□□□□□□ “□□”
3. □□□□□
4. □□ “□□□□”
5. □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ □ □ □ □

- 88888888
- 88888888
- 8888888
- 2FA 88888888

□ □ □ □

[illegible]

□□□□□□

□□□□□□

- □□□□ - □□□□□□□□□□“□□ - +44 7700 900123”□
- □□ - □□□□□□□
- □□ - □□□□□□□□□□
- □□□□ - □□□□□□
- □□□□ - □□□□□□□□□□
- □□ - □□□□
- □□□□ - □□/□□□□

□□□□□

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

□□□□□□□□

- □ □□ - □□□□□□
- □ □□□□ - □<7□□□□□□□
- □ □□ - □□□□□□□□□□□□□□□□
- ● □□ - □□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□□□

□□□□

□□□□□□□□□□□□□□□□□□

பொருள்

- பொருள் - பொருள்   
- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்
- பொருள் - பொருள்

பொருள்

பொருள் + பொருள்

- பொருள் 10 - பொருள்/பொருள்
- பொருள் 20 - பொருள்
- பொருள் 30 - பொருள்

பொருள்

பொருள்

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 - 00000
- 00**MMS** - MMS000000
- 0000 - 00000000000000000000

0000

00000000000000000000

00000

- 0000 - 0000
- 0000 - 000000000000
- 0000 - 000000000000
- 0000 - 00/0000

0000

00000000000000000000

00

00000000000000000000

□□□□□□

- □□□□ - □□□□□
- □□ - □□□□□□
- □□□□ - □□□□□□
- □□ - □□□□□
- □□ - □□□□□□□□□□
- □□ - □□PDF□□□□□

□□□□□

1. □□□ □□ → □□
2. □□□□□□□□□□□□
3. □□ “□□□□” □□
4. □□□□□□□□□□□□□□
5. □□□□
6. □□□□□□□□

□□□□□

1. □□□□□□ “□□” □□

2. PDF

3.

- -
- - /
- -
- -
- -

- - Stripe
- -
- -
- - 4

/

1. → → “”
- 2.

3. 選擇合適的數據中心
4. 選擇合適的服務器
5. 選擇合適的網絡
6. 選擇合適的軟件
7. 選擇合適的硬件

選擇合適的服務器

- 服務器 - 1GB、5GB、10GB、20GB、50GB
- 服務器 - 數據中心
- 服務器 - 網絡
- 服務器 - 軟件、£5、£10、£20、£50、£100

選擇合適的網絡 & 數據中心

選擇合適的服務器

選擇合適的服務器

選擇合適的服務器

- 服務器 - 數據中心
- 服務器 - IP、數據中心、IP
- 服務器 - IPTV、數據中心、IP
- 服務器 - 數據中心、數據中心
- 服務器 - 數據中心、數據中心

選擇合適的服務器

1. 選擇合適的服務器 → 選擇合適的服務器 → “選擇合適的服務器”
2. 選擇合適的服務器
3. 選擇合適的服務器 “選擇合適的服務器”
4. 選擇合適的服務器 + 選擇合適的服務器
5. 選擇合適的服務器
6. 選擇合適的服務器

【重要】 2025 - 2026 年度の試験日程について

試験概要

試験科目と試験時間

試験形式

- 筆記試験
- 口頭試験
- 面接試験(7月3日1時)
- 書類審査(書類提出)
- 英語/国語
- 数学
- 面接試験(書類提出)

試験会場

- 東京会場
- 大阪会場
- 名古屋会場(10%)
- 福岡会場
- 札幌会場(書類提出)

試験料

試験結果

- 合格発表
- 不合格発表
- 不合格理由(??) 不合格<20%の場合
- 再試験

□□□□□□

□□□□□□□□

□□□□

- □□□□□□□□ ↔ □□□
- □□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□
- □□□□
- □□□□□□□
- □□□□□□□

□□□□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□
- □□□□□□□□□□□□□□□□/□□□

□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

1. □□□ □□ → □□□□
2. □□ □□□ □□□
3. □□□□□□□□□□□□□□
4. □□ “□□□□□□□□” □□
5. □□□□□□□□□□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□
7. □□□□□□□□□□□□□□□□□□

□□□

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

□□□□□□

- □□ `can_impersonate` □ `admin` □□
- □□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□□□

□□□ □□□□ - □□□□ □□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□□□

- 網路連線
- 網路連線
- 網路連線
- 網路連線
- 網路eSIM網路連線
- 網路連線
- 網路連線GPS

WebPWA

- 網路連線
- 網路連線
- 網路連線
- 網路連線

網路連線

網路連線網路連線網路連線

網路連線

1. 網路連線 “網路”
2. 網路連線
3. 網路連線網路連線1
4. 網路連線
5. 網路連線網路連線
6. 網路連線

網路

- 8
- 網路
- 網路
- 網路
- 網路
- 網路!@#\$%^&*

00000

5000000000

- 00003000
- 000000000000
- 000000000000

0000000

00000

1. 002FA000000
2. 0000000000
3. 00000000000000
4. 00000000000000
5. 00000000000000
6. 00000000000000002FA
7. 00000/00000000

00000

1. 000000000010-20%
2. 00000000000000
3. 000000000000
4. 000000000000
5. 000000000000000000

00000

000000000000

1. 00000/0000000000
2. 00000000000000
3. 0000000000000000
4. 0000ID000000000000

API

API

API

API

1. API
2. “API”
3. API
4. API
5. API

API

- API
- API
- API
- API
- API

API

API

API

API

API

API

API

API

FAQ

FAQ 常见问题解答

FAQ 常见问题解答

FAQ

FAQ 常见问题解答

1. 常见问题解答
2. 常见问题解答
3. 常见问题解答
4. 常见问题解答
5. 常见问题解答
6. 常见问题解答
7. 常见问题解答

FAQ

FAQ 常见问题解答

- 常见问题解答
- 常见问题解答 Markdown 常见问题解答
- 常见问题解答
- 常见问题解答
- 常见问题解答
- 常见问题解答

FAQ

- 常见问题解答
- 常见问题解答

- 如何選擇合適的保險公司
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人

FAQ

- 如何選擇合適的保險公司
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人

FAQ

FAQ

- 如何選擇合適的保險公司
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人
- 如何選擇合適的保險計劃

FAQ

- 如何選擇合適的保險公司
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人
- 如何選擇合適的保險計劃
- 如何選擇合適的保險代理人

FAQ

如何選擇合適的保險公司

如何選擇合適的保險公司

000

□ □ □ □ □ □ □ □

Markdown HTML

1 Markdown

- Markdown HTML
-
- Markdown
- /
-

2

Web UI

1. →
- 2.
3.
 -
 - Markdown
 -
 -
 -
- 4.

API

POST /crm/faq/

create_faq

```
{
  "question": " ",
  "answer": "1. 2. 3. 4. 5. ",
  "category": "",
  "tags": "",
  "is_visible_to_customer": true
}
```

```
{
  "faq_id": 42,
  "question": " ",
  "answer": "1. ...",
  "category": "",
  "tags": "",
  "is_visible_to_customer": true,
  "view_count": 0,
  "helpful_count": 0,
  "not_helpful_count": 0,
  "created": "2026-01-07T10:00:00Z",
  "last_modified": "2026-01-07T10:00:00Z",
  "created_by_user": {
    "id": 1,
    "username": "admin",
    "first_name": "Admin",
    "last_name": "User"
  }
}
```

目錄

第一章

1. 第一章
2. 第二章
3. 第三章
4. 第四章

第二章

- 第二章
- 第三章
- 第四章

第三章

第四章

1. 第四章
2. 第五章
3. 第六章
 - 第七章
 - 第八章
4. 第九章

第五章

- 第五章
- 第六章
- 第七章
- 第八章

FAQ 問答

問問問

問問問問問問問問問

- 問問問問問問問問問問
- 問問問問問問問問
- 問問問問問問問問問問
- 問問問問問問問問
- 問問問SIM 卡❓❓問問問問
- 問問問問問問問問問問
- **VoIP** 問問問問問問問問問
- **IPTV** 問問問問問問問問問

問問問

問問問問問問問問問問問問問問問問

1. 問問問問問問
2. 問問問問
3. 問問問問問問
4. 問問問/問問問

問問問

問問問問問

GET /crm/faq/categories

問問問問問問問問問問問問問問

問問問問


```
GET /crm/faq/customer/categories
```

□ □ □ □ □ □ □ □ □ □ □ □

11

11

□ □ □ □ □ □ □ □ □

- 
- 
- 
- 

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

- 0000 00000000bug000000
- 0000 0000000000000000
- 0000 0000000000000000
- 0000 000000000000
- 0000 0000000000000000

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

1. 0000000000000000
2. 0000000"password-reset"000000"password"
3. 0000000000000000??
4. 0000000000000000 3-5 000
5. 0000000000000000

問問 FAQ 問問

問問問問問

問問問問問問問問問問

1. 問問問問問問問問問問
2. 問問問問問問“問問問問問問問問”
3. 問問問問問問問問問問
4. 問問問問問問問問問問

問問問問問

問問問問問

- 問問問問問問問/問問
- 問問問問問問問
- 問問問問問問問問問問
- 問問問問問
- 問問問問問問/問問問
- 問問問問問問問問問問問問

問問問問

問問問問

- 問問問問問問問問問問問問問問
- 問問問問問問問問問問問問問問
- 問問問問問問問問問問問問問問
- 問問問問問問問問問問問問問問
- 問問問問問問問問問問問問

問問問問問

問問問問問

1. 註冊帳號
2. 登入系統
3. 選擇合適的課程
4. 開始學習
5. 完成課程

常見問題

- 如何註冊帳號
- 如何登入系統
- 如何選擇課程
- 如何開始學習 + 如何完成課程

FAQ

常見問題

如何註冊帳號

- 如何註冊帳號
- 如何登入系統
- 如何選擇課程
- 如何開始學習
- 如何完成課程

常見問題

常見問題

- 如何註冊帳號
- 如何登入系統
- 如何選擇課程
- 如何開始學習
- 如何完成課程

FAQ 問答

問問問問問問問問

問問問問

- 問問問問問問問問
- 問問問問問問問問
- 問問問問問問問問
- 問問問問問問問問

問問問問

- 問問問問問問問問
- 問問問問問問問問
- 問問問問問問問問
- 問問/問問問問問問問問

API 問答

GET /crm/faq/popular?limit=10
GET /crm/faq/helpful?limit=10

問問問問

問問問問問問

1. 問問問問問問問問問問
2. 問問問問問問問問問問問問
3. 問問❓❓問問問問問問問問問問
4. 問問問問問問問問問問問問
5. 問問問問問問問問問問問問問問

Markdown 問答

問問問問問問 Markdown 問問問問問問問問

□□□□

```
**□□□□**  □□□□
*□□□□*  □□□□□□
~~□□□~~  □□□□□□
`□□□□`  □□□□□□
```

□□

```
##  □□□□
### □□□□□
#### □□□
```

□□

```
□□□□□
1.  □□□
2.  □□□
3.  □□□
```

```
□□□□□
-  □□□
-  □□□
-  □□□
```

□□

```
[□□□□](https://example.com)
[□□□□□□](mailto:support@example.com)
```

□□□

```
```bash
command --with-flags
```

```json
{
 "example": "data"
}
```
```

□□

```
! [□□□□] (https://example.com/image.png)
```

□□

```
| □ 1   | □ 2   |
|-----|-----|
| □□ 1 | □□ 2 |
```

□□□

```
> □□□□□□□
> □□□□
```

API □□

□□□□□□□□□□□□

```
GET /crm/faq/
```

view_faq

- page 1
- per_page 20
- sort created, view_count, helpful_count
- order asc, desc
- search
- category
- tags
- is_visible_to_customer true/false

GET /crm/faq/customer

GET /crm/faq/faq_id/{faq_id}

GET /crm/faq/customer/faq_id/{faq_id}

□□□□□□

```
PATCH /crm/faq/faq_id/{faq_id}
```

□□□□□ `update_faq`

□□□□□□

```
DELETE /crm/faq/faq_id/{faq_id}
```

□□□□□ `delete_faq`

□□□□□□□□□□□□□□□□□□

□□□□□□

```
POST /crm/faq/faq_id/{faq_id}/visibility
```

□□□□□ `manage_faq_visibility`

□□□

```
{
  "is_visible_to_customer": true
}
```

□□□□□□□□□□

```
POST /crm/faq/faq_id/{faq_id}/helpful
```

□□□□□□□ □ □□□   □□□□□□□□□□

□□□


```
{  
  "is_helpful": true  
}
```

□□□□□□□□

```
GET /crm/faq/popular?limit=10
```

□□□□□□□□□□□□□□

□□□□□□□□□□□□

```
GET /crm/faq/helpful?limit=10
```

□□□□□□□□□□□□□□

□□□□

□□□□

1. □□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□
3. □□□□□□□□□□□□
4. □□□□□□□□□□□□□□
5. □□□□□□□□□□□□

□□

1. □□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□
4. □□□□□□□□□□□□□□

5. 如何選擇合適的數據庫

問題

1. 數據庫的選擇標準
2. 數據庫的選擇過程
3. 數據庫的選擇結果
4. 數據庫的選擇總結
5. 數據庫的選擇建議

答案

數據庫的選擇標準

- 數據庫的選擇標準“數據庫的選擇標準”
- 數據庫的選擇標準“數據庫的選擇標準”
- 數據庫的選擇標準“數據庫的選擇標準”

數據庫的選擇過程

- 數據庫的選擇過程“數據庫的選擇過程”
- 數據庫的選擇過程“數據庫的選擇過程”
- 數據庫的選擇過程“數據庫的選擇過程”

數據庫的選擇結果

- 數據庫的選擇結果“數據庫的選擇結果”
- 數據庫的選擇結果“數據庫的選擇結果”
- 數據庫的選擇結果“數據庫的選擇結果”

數據庫的選擇總結

- 數據庫的選擇總結“數據庫的選擇總結” `update_faq`
- 數據庫的選擇總結“數據庫的選擇總結”
- 數據庫的選擇總結“數據庫的選擇總結”



OmniCRM 系統整合專案進度追蹤表

系統名稱: 客戶關係管理系統 (CRM) | 專案編號: CRM-2024-001 | 負責人: 張三

1. 專案背景與目標



目標說明:

1. 系統整合與數據同步
2. 提升客戶服務效率
3. 加強銷售團隊協作
4. 實現客戶生命周期管理
5. **SLA** 達成率提升
6. 系統穩定性與安全性
7. 用戶培訓與支持



□□□□

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

□□

- □□□□□□□□□□□□□□□□
- □□□□□□□ OmniCRM □□□□□□□
- @□□□□□ @□□□□ □□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□

□□□□

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

專案需求

1. 目標

Web UI

1. 首頁 → 登入
2. 登入成功
3. 登入失敗
 - 帳號密碼錯誤
 - 帳號密碼錯誤
 - 帳號密碼錯誤
 - 帳號密碼錯誤
 - 帳號密碼錯誤
 - 帳號密碼錯誤
 - 帳號密碼錯誤
4. 登入成功

登入成功

登入成功後顯示的內容

- 登入成功
- 登入成功
- 登入成功
- 登入成功

API

POST /crm/ticket/

登入

```
{
  "title": "🔧🔧🔧🔧🔧",
  "description": "🔧🔧🔧🔧🔧🔧🔧🔧🔧🔧🔧",
  "customer_id": 290,
  "service_id": 167,
  "priority": "High",
  "status": "Open",
  "tags": "technical,voicemail"
}
```

2. 📧📧📧

📧📧📧📧📧📧📧📧📧

1. 📧📧📧📧📧“📧”📧📧📧📧
2. 📧📧📧📧📧📧📧📧📧📧📧📧📧
3. 📧📧📧📧📧📧📧📧📧
4. 📧📧📧📧🔧🔧🔧🔧📧📧📧📧📧📧📧📧

📧📧📧📧📧📧

- 📧📧📧📧📧📧📧📧📧
- 📧📧📧📧📧📧📧
- 📧📧📧📧📧📧“📧📧📧”📧📧

3. 📧📧📧

📧📧📧📧

1. 📧📧📧
2. 📧📧 📧📧📧 📧📧📧📧
3. 📧📧📧📧📧
4. 📧📧 @📧📧📧 📧📧📧📧📧📧📧📧📧@john 📧📧📧📧📧📧
5. 📧📧📧📧📧
 - 📧📧📧📧📧📧📧📧📧
 - 📧📧📧📧📧📧📧📧📧📧📧
6. 📧📧 📧📧

@

@

-
-
-
-

1. 10MB
2. PDF
- 3.
- 4.

-
-
-
-
-

- SLA 1
- SLA 4
- SLA 24
- SLA 48

4.

- 1.
- 2.

3

-
-
-

- ID**
-
-
-
-

- 提供IPTV及VoIP
- 提供
 - 网络/电话
 - 网络/电视
 - 网络/网络

网络

- 提供TKT-2026-00042
- 提供
- 提供
- 提供
- 提供
- 提供
- 提供

网络

网络

- 提供
- 提供
- 网络/网络

网络

网络

网络

1. 提供
2. 提供
3. 提供
4. 提供

5. 0000000000000000

6. 00000000000000

🔍🔍🔍00000000

- From00000000
- Subject000000000000000000
- In-Reply-To00000000
- Message-ID00000000

0000

00000000000000

- 00000000000000
- 00000000000000
- 00000000000000

00000000

- {{ticket_number}}0000
- {{customer_name}}0000
- {{update_text}}0000
- {{staff_name}}00000000
- {{ticket_url}}00000000

0000000

0000000000000000

1. 00000000
2. 0000“00”0“0000”
3. 00000000000000
4. 000000000000000000000000
5. 00000000000000
6. 0000000000

7. 環境問題の取り組み

環境問題

- 環境問題の現状
- 環境問題の原因
- 環境問題の影響
- 環境問題の対策

環境問題

環境問題

環境問題

- 環境問題の現状
- 環境問題の原因
- 環境問題の影響 @環境 環境問題

環境問題

環境問題

1. 環境問題
2. 環境問題の原因
3. 環境問題の影響
4. 環境問題の対策
5. 環境問題

環境問題

1. 環境問題の原因 X
2. 環境問題
3. 環境問題の原因

□□□□

□□□□□□□□□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- @□□□□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□□

□□□□

□□□□□□□

- □□□□□□
- □□□□
- □□□□
- □□□□
- □□□□

□□□□

□□□□□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□
- □□□□□□□
- □□□□□□□□□
- □□□□□□□□□□□□
- □□□□□/□□□□□□□

目標設定

目標設定の重要性

1. 目標設定はモチベーションを高める
2. 目標設定は“何をやるべきか”を明確にする
3. 目標設定は進捗管理を容易にする
4. 目標設定は責任感を促す

目標設定の方法

- “SMART”原則に従って目標を設定する
- “短期目標”と“長期目標”を明確にする
- “目標達成”のための具体的な計画を立てる
- “目標達成”のための7つのステップ

モチベーション

モチベーション

モチベーションの源泉

- 目標設定によるモチベーション
- 報酬によるモチベーション
- 承認によるモチベーション
- 成長によるモチベーション
- 自己効力感によるモチベーション vs 他 vs 他 vs 他
- 環境によるモチベーション
- 習慣によるモチベーション

SLA 概要

SLA の重要性

- SLA の種類と特徴

- **SLA** 服務等級協議
- 服務等級 SLA 服務協議
- 服務等級協議服務協議

服務

服務等級協議

- 服務等級協議服務協議
- 服務等級協議服務協議
- 服務等級協議服務協議
- 服務等級協議服務協議

服務等級協議

服務等級協議

服務等級協議

1. 服務等級協議服務協議
2. 服務等級協議服務協議
3. 服務等級協議服務協議
4. 服務等級協議服務協議

服務等級協議

服務等級協議

服務等級協議

1. 服務等級協議 服務
2. 服務等級協議
3. 服務等級協議
4. 服務等級協議
5. 服務等級協議

API

1. API 是什麼?
2. API 的用途是什麼?
3. API 的種類有哪些?
4. API 的開發流程是什麼?

API

1. API 的定義
2. API 的用途
3. API 的種類
4. API 的開發
5. API 的測試

API

- API 是什麼?
- @API 是什麼?
- API 的 API 是什麼?
- API 的 API

API

- API 是什麼 - API 是什麼?
- API 是什麼?
- API 是什麼?

API

- @API 是什麼?
- API 是什麼?
- API 是什麼?

API API 是什麼?

API 接口

接口列表

POST /crm/ticket/

接口名称 create_ticket

接口描述

GET /crm/ticket/ticket_id/{ticket_id}

接口名称 view_ticket

接口描述

PATCH /crm/ticket/ticket_id/{ticket_id}

接口名称 update_ticket

接口描述

DELETE /crm/ticket/ticket_id/{ticket_id}

接口名称 delete_ticket

接口描述

POST /crm/ticket/ticket_id/{ticket_id}/updates

接口

POST /crm/ticket/ticket_id/{ticket_id}/attachments

接口名称 upload_attachment

接口

POST /crm/ticket/ticket_id/{ticket_id}/subscribe

接口名称 subscribe_ticket

接口

接口说明

- 接口功能
- 接口地址
- 接口参数

接口参数

- 接口参数
- 接口返回
- 接口说明

接口说明

- 接口说明
- 接口地址
- 接口参数

接口说明

- 接口说明

- 00000000000000000000000000000000
- 00000000000000000000000000000000