Odoo Clover Terminal Connector (sample module)

This repository is a **starter** Odoo module that demonstrates how to connect Odoo (POS / Invoicing) with the Clover Terminal APIs (REST Pay / Clover REST). It includes OAuth, a payment request flow, webhook handling for payment results, and a small POS extension to send the payment request to Clover.

> **IMPORTANT**: This is a sample integration to get you started. You must test it in a sandbox Clover account, add proper error handling, secure storage of secrets, and adapt to your Odoo version (the code targets Odoo 15/16 style patterns).

File tree

```
odoo_clover_terminal/
├── __init__.py
├── __manifest__.py
├── README.md
├── controllers/
│   └── main.py
├── models/
│   ├── __init__.py
│   ├── clover_transaction.py
│   └── res_config_settings.py
├── views/
│   ├── clover_settings_views.xml
│   └── clover_webhook_views.xml
├── static/
│   └── src/
│       ├── js/
│       │   └── pos_clover.js
│       └── xml/
│           └── pos_clover_templates.xml
└── security/
    └── ir.model.access.csv
```

`__manifest__.py`

```
{
    "name": "Clover Terminal Connector",
    "version": "0.1",
    "summary": "Connect Odoo to Clover Terminals (pay via Clover REST/REST Pay)",
```

```
    "category": "Accounting/Point of Sale",
    "author": "Your Company",
    "depends": ["base", "web", "point_of_sale", "payment"],
    "data": [
        "views/clover_settings_views.xml",
        "views/clover_webhook_views.xml",
        "security/ir.model.access.csv",
    ],
    "qweb": [
        "static/src/xml/pos_clover_templates.xml",
    ],
    "assets": {
        "point_of_sale.assets": [
            "odoo_clover_terminal/static/src/js/pos_clover.js",
        ],
    },
    "installable": True,
    "application": False,
}
```

`__init__.py`

```
from . import models
from . import controllers
```

`models/__init__.py`

```
from . import clover_transaction
from . import res_config_settings
```

`models/res_config_settings.py` (store Clover credentials in system parameters via settings)

```
from odoo import models, fields

class ResConfigSettings(models.TransientModel):
    _inherit = 'res.config.settings'

    clover_client_id = fields.Char(string='Clover Client ID')
    clover_client_secret = fields.Char(string='Clover Client Secret')
    clover_redirect_uri = fields.Char(string='Clover OAuth Redirect URI')
    clover_merchant_id = fields.Char(string='Clover Merchant ID')
    clover_api_base = fields.Char(string='Clover API Base URL',
```

```
default='https://api.clover.com')

    def set_values(self):
        super().set_values()
        params = self.env['ir.config_parameter'].sudo()
        params.set_param('clover.client_id', self.clover_client_id or '')
        params.set_param('clover.client_secret', self.clover_client_secret or
''')
        params.set_param('clover.redirect_uri', self.clover_redirect_uri or
'')
        params.set_param('clover.merchant_id', self.clover_merchant_id or '')
        params.set_param('clover.api_base', self.clover_api_base or 'https://
api.clover.com')

    def get_values(self):
        res = super().get_values()
        params = self.env['ir.config_parameter'].sudo()
        res.update({
            'clover_client_id': params.get_param('clover.client_id',
default=''),
            'clover_client_secret': params.get_param('clover.client_secret',
default=''),
            'clover_redirect_uri': params.get_param('clover.redirect_uri',
default=''),
            'clover_merchant_id': params.get_param('clover.merchant_id',
default=''),
            'clover_api_base': params.get_param('clover.api_base',
default='https://api.clover.com'),
        })
        return res
```

---

models/clover_transaction.py

```
from odoo import models, fields, api
import logging
import requests

_logger = logging.getLogger(__name__)

class CloverTransaction(models.Model):
    _name = 'clover.transaction'
    _description = 'Clover Transaction'

    name = fields.Char(string='Reference', readonly=True)
    odoo_order_id = fields.Many2one('pos.order', string='POS Order')
    clover_payment_id = fields.Char(string='Clover Payment ID')
    amount = fields.Float(string='Amount')
    currency = fields.Char(string='Currency', default='USD')
```

```
    status = fields.Selection([('pending','Pending'), ('paid','Paid'),
('failed','Failed')], default='pending')
    data = fields.Json(string='Raw Data')

    def create_payment_request(self, amount_cents, order_reference):
        """
        Use Clover REST Pay Display API to create a payment request on a
nearby Clover device.
        This is a simplified synchronous example — consider async flows for
production.
        """
        params = self.env['ir.config_parameter'].sudo()
        api_base = params.get_param('clover.api_base') or 'https://
api.clover.com'
        merchant_id = params.get_param('clover.merchant_id')
        access_token = params.get_param('clover.access_token')  # You must
obtain and store OAuth token

        if not access_token:
            raise Exception('Clover access token missing in system
parameters')

        url = f"{api_base}/v3/merchants/{merchant_id}/pay"
        # REST Pay Display expects a POST. Payload fields depend on API
version. This is an illustrative example.
        payload = {
            'amount': int(amount_cents),
            'currency': 'USD',
            'externalPaymentId': order_reference,
            # more fields: note, autoCapture, tipAmount, etc.
        }
        headers = {
            'Authorization': f'Bearer {access_token}',
            'Content-Type': 'application/json'
        }
        _logger.info('Sending payment request to Clover: %s', payload)
        r = requests.post(url, json=payload, headers=headers, timeout=30)
        if r.status_code in (200,201):
            res = r.json()
            self.write({'status':'pending', 'data': res, 'clover_payment_id':
res.get('id')})
            return res
        else:
            _logger.error('Clover payment request failed: %s %s',
r.status_code, r.text)
            raise Exception(f'Failed to create Clover payment:
{r.status_code} {r.text}')
```

controllers/main.py

```
from odoo import http
from odoo.http import request
import logging
import requests

_logger = logging.getLogger(__name__)

class CloverController(http.Controller):

    @http.route(['/clover/oauth/callback'], type='http', auth='public',
csrf=False)
    def clover_oauth_callback(self, **kwargs):
        """Handle Clover OAuth callback and store token in system params
(example)."""
        code = kwargs.get('code')
        error = kwargs.get('error')
        params = request.env['ir.config_parameter'].sudo()
        client_id = params.get_param('clover.client_id')
        client_secret = params.get_param('clover.client_secret')
        redirect_uri = params.get_param('clover.redirect_uri')
        api_base = params.get_param('clover.api_base') or 'https://
www.clover.com'

        if error:
            return "OAuth error: %s" % error
        if not code:
            return "Missing code"

        token_url = api_base.rstrip('/') + '/oauth/token'
        resp = requests.post(token_url, data={
            'grant_type': 'authorization_code',
            'code': code,
            'client_id': client_id,
            'client_secret': client_secret,
            'redirect_uri': redirect_uri,
        })
        if resp.status_code != 200:
            _logger.error('Failed to exchange OAuth code: %s', resp.text)
            return 'OAuth token exchange failed'
        data = resp.json()
        access_token = data.get('access_token')
        refresh_token = data.get('refresh_token')
        merchant_id = data.get('merchant_id') or
params.get_param('clover.merchant_id')

        # Save tokens (for demo: in system params). In production, store
securely.
        params.set_param('clover.access_token', access_token)
```

```
        params.set_param('clover.refresh_token', refresh_token)
        params.set_param('clover.merchant_id', merchant_id)

        return 'Clover connected successfully. You may close this window.'

    @http.route(['/clover/webhook'], type='json', auth='none', csrf=False)
    def clover_webhook(self, **post):
        """Receive webhooks from Clover (payments update) and update
clover.transaction records."""
        # You should verify signature if Clover provides one.
        data = request.httprequest.get_json(force=True)
        _logger.info('Received Clover webhook: %s', data)
        external_id = data.get('externalPaymentId') or data.get('id')
        payment_id = data.get('id')
        status = data.get('status')
        amount = data.get('amount')

        tx =
request.env['clover.transaction'].sudo().search([('clover_payment_id','=',payment_id)],
limit=1)
        if not tx and external_id:
            # Try to find by externalPaymentId stored as name
            tx =
request.env['clover.transaction'].sudo().search([('name','=',external_id)],
limit=1)
        if tx:
            vals = {'data': data}
            if status and status.lower() in ('paid','completed'):
                vals['status'] = 'paid'
            elif status and status.lower() in ('failed','voided'):
                vals['status'] = 'failed'

request.env['clover.transaction'].sudo().browse(tx.id).write(vals)
        else:
            # Create a new record if not found (optional)
            request.env['clover.transaction'].sudo().create({
                'name': external_id or payment_id,
                'clover_payment_id': payment_id,
                'amount': (amount or 0)/100.0,
                'status': 'paid' if (status and status.lower()=='paid') else
'pending',
                'data': data,
            })
        return { 'ok': True }
```

`views/clover_settings_views.xml`

```
<odoo>
  <record id="view_clover_settings" model="ir.ui.view">
    <field name="name">res.config.settings.clover.form</field>
    <field name="model">res.config.settings</field>
    <field name="arch" type="xml">
      <form string="Clover Settings">
        <sheet>
          <group>
            <field name="clover_client_id"/>
            <field name="clover_client_secret"/>
            <field name="clover_redirect_uri"/>
            <field name="clover_merchant_id"/>
            <field name="clover_api_base"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>
</odoo>
```

`views/clover_webhook_views.xml` (simple tree to view transactions)

```
<odoo>
  <record id="view_clover_transaction_tree" model="ir.ui.view">
    <field name="name">clover.transaction.tree</field>
    <field name="model">clover.transaction</field>
    <field name="arch" type="xml">
      <tree>
        <field name="name"/>
        <field name="odoo_order_id"/>
        <field name="amount"/>
        <field name="status"/>
      </tree>
    </field>
  </record>

  <record id="action_clover_transactions" model="ir.actions.act_window">
    <field name="name">Clover Transactions</field>
    <field name="res_model">clover.transaction</field>
    <field name="view_mode">tree,form</field>
  </record>

  <menuitem id="menu_clover_root" name="Clover"/>
```

```
  <menuitem id="menu_clover_transactions" name="Transactions"
parent="menu_clover_root" action="action_clover_transactions"/>
</odoo>
```

`static/src/js/pos_clover.js` (POS client side extension)

```
odoo.define('odoo_clover_terminal.pos_clover', function(require){
    'use strict';
    const screens = require('point_of_sale.screens');
    const rpc = require('web.rpc');

    const PosModel = require('point_of_sale.PosModel');
    const models = require('point_of_sale.models');

    // Simple function to call server to create an entry and then request
Clover payment
    const _super_order = models.Order.prototype.export_for_printing;
    models.Order = models.Order.extend({
        export_for_printing: function(){
            return _super_order.apply(this, arguments);
        }
    });

    // Add a new payment method action in the payment screen (conceptual)
    const PaymentScreen = screens.PaymentScreenWidget;
    screens.PaymentScreenWidget.include({
        click_numpad: function(){
            // keep default behaviour
            this._super.apply(this, arguments);
        },
        pay_with_clover: async function(amount){
            // Call Odoo controller to create clover.transaction and ask
server to call Clover REST
            const order = this.pos.get_order();
            const payload = {
                amount_cents: Math.round(amount*100),
                order_reference: order.uid,
            };
            const res = await rpc.query({
                model: 'clover.transaction',
                method: 'create_payment_request',
                args: [payload.amount_cents, payload.order_reference],
            });
            // After request, the webhook (server) will update transaction
status
            this.gui.show_popup('confirm',{
                'title': 'Clover payment requested',
                'body': 'Please complete the payment on the Clover device.'
```

```
            });
        }
    });
});
```

`static/src/xml/pos_clover_templates.xml`

```xml
<?xml version="1.0"?>
<t t-name="odoo_clover_terminal.pos_clover_templates">
  <!-- create UI components if needed -->
</t>
```

`security/ir.model.access.csv`

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_clover_transaction,access_clover_transaction,model_clover_transaction,,
1,1,1,1
```

How it works (high level)

1. Admin configures Clover OAuth credentials in Settings (Res Config Settings) and performs OAuth flow to store access token.
2. POS user clicks a custom "Pay with Clover" action which calls server method `clover.transaction.create_payment_request`.
3. Server calls Clover REST Pay or Payments API to create a payment request on the merchant's Clover device.
4. Clover device processes the payment. Clover then triggers a webhook to `/clover/webhook` with payment status.
5. The controller updates the `clover.transaction` record and you can reconcile the POS order as paid.

Notes & Next steps

- **OAuth & token refresh:** This example stores tokens in system parameters. Use a secure store and implement refresh token flow (Clover returns refresh_token).
- **API paths & payloads:** Clover has different endpoints (Developer Pay, REST Pay Display, Android Payments). Inspect the Clover docs and adjust payloads accordingly: https://docs.clover.com/dev/docs/rest-pay-intro. (Links in module README recommended).
- **Signature verification:** If Clover signs webhooks, validate signatures to avoid spoofing.
- **Error handling & retries:** Add robust error handling and logging.
- **PCI/Compliance:** Be mindful of PCI rules: do not log sensitive card data.

- **Testing:** Use Clover sandbox / developer environment for testing.

If you want, I can:

- Convert this into a downloadable zip you can install into Odoo (I can generate the files here).
- Expand the POS UI/UX to add a dedicated payment button and auto-reconcile order when webhook marks payment paid.
- Implement OAuth token-refresh and better secure storage (using a model and encryption).

Tell me which of those you'd like next and I will produce it.