OCPP & UPI mobile payments

v1.0, October 2025

Table of Contents

Author: Open Charge Alliance

1. Introduction

Digital payments are rapidly becoming the preferred method for everyday transactions around the world. In many countries, the rise of mobile-first payment systems has created opportunities to make public EV charging more accessible and convenient without the need for cards, proprietary apps, or RFID tokens.

For Charge Point Operators (CPOs), integrating these mobile payment systems is not just about offering another way to pay. The design of the integration has a direct impact on how seamless the driver experience will be, how efficiently charging sessions can be started and stopped, and how reliably the operator can reconcile payments.

Two of the most prominent examples of mobile-first payment ecosystems are:

- **UPI (Unified Payments Interface)** in India, which enables instant account-to-account payments using QR codes or mobile apps.
- **M-PESA** in Kenya and other African countries, which allows users to send and receive money through their mobile phones without requiring a bank account.

Although the underlying technologies differ, both systems present similar integration challenges for EV charging. Operators must decide how to connect payment flows to their Charging Station Management System (CSMS), how to link transactions to specific chargers, and how to ensure that prepaid limits or refunds are handled correctly.

This paper explores how UPI can be integrated into EV charging payments, looking at both simple and advanced models of integration. It also highlights how the Open Charge Point Protocol (OCPP) supports these scenarios through features such as remote transaction control and prepaid transaction limits.

OCA have published a similar paper on M-PESA payments, which can be found on the openchargealliance.org website.

2. Using UPI for EV charging payments: static vs. dynamic integration

The Unified Payments Interface (UPI) has transformed digital transactions in India, offering instant, 24/7 payments directly between bank accounts. This capability makes it a natural fit for public EV charging, where drivers can pay quickly without the friction of cards, wallets, or cash.

For Charge Point Operators (CPOs), integrating UPI payments requires more than simply printing a QR code on a charger. The way the UPI payment server interacts with the CPO backend determines how automated, reliable, and user-friendly the process will be.

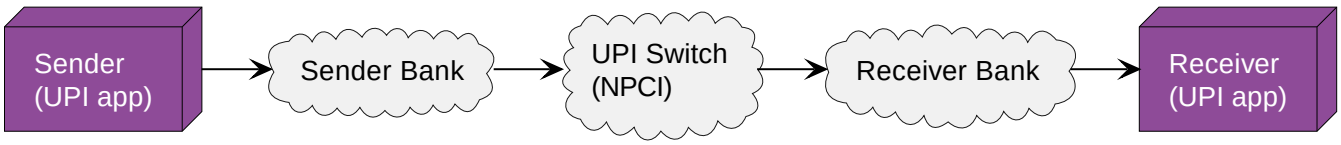

Figure 1. UPI payment flow

2.1. Two integration models for UPI payments

When implementing UPI for EV charging, CPOs typically choose between **Pure UPI static QR code** and **Webpage-redirect QR code** flows.

2.1.1. Pure UPI static QR code

In the static model, the charger has a fixed UPI QR code (as a sticker or on a display) linked to the CPO's merchant VPA (for example, `pa=cpo123@hdfcbank`) and a transaction reference or note that identifies the charger (for example, `tr=EVSE1234-1`). The driver scans this code with their preferred UPI-enabled app, enters the prepaid amount and the payment is routed from the user's bank, through the NPCI UPI switch, to the CPO's acquiring bank. Once the payment appears in the CPO's account, the operator or their system sends out an OCPP `RemoteStartTransaction` (OCPP 1.6) or `RequestStartTransaction` (OCPP 2.x) to the charger that was referenced in the UPI QR code. The operator will have to ensure that the transaction is stopped before the prepaid amount is exceeded. Detais of this are discussed in section How OCPP supports prepaid payments.

The payment request as a result of scanning the UPI QR code looks like this:

```
upi://pay?pa=cpo123@hdfcbank&pn=CPO-Name&tr=EVSE1234-1&am=200&cu=INR
```

The main advantage of this approach is its simplicity. It is inexpensive to deploy and works without requiring a mobile app from the operator, which makes it appealing for small-scale operators. However, this works with prepaid amounts, and refunds, when a session fails to start or when the full prepaid amount is not used, are not possible.

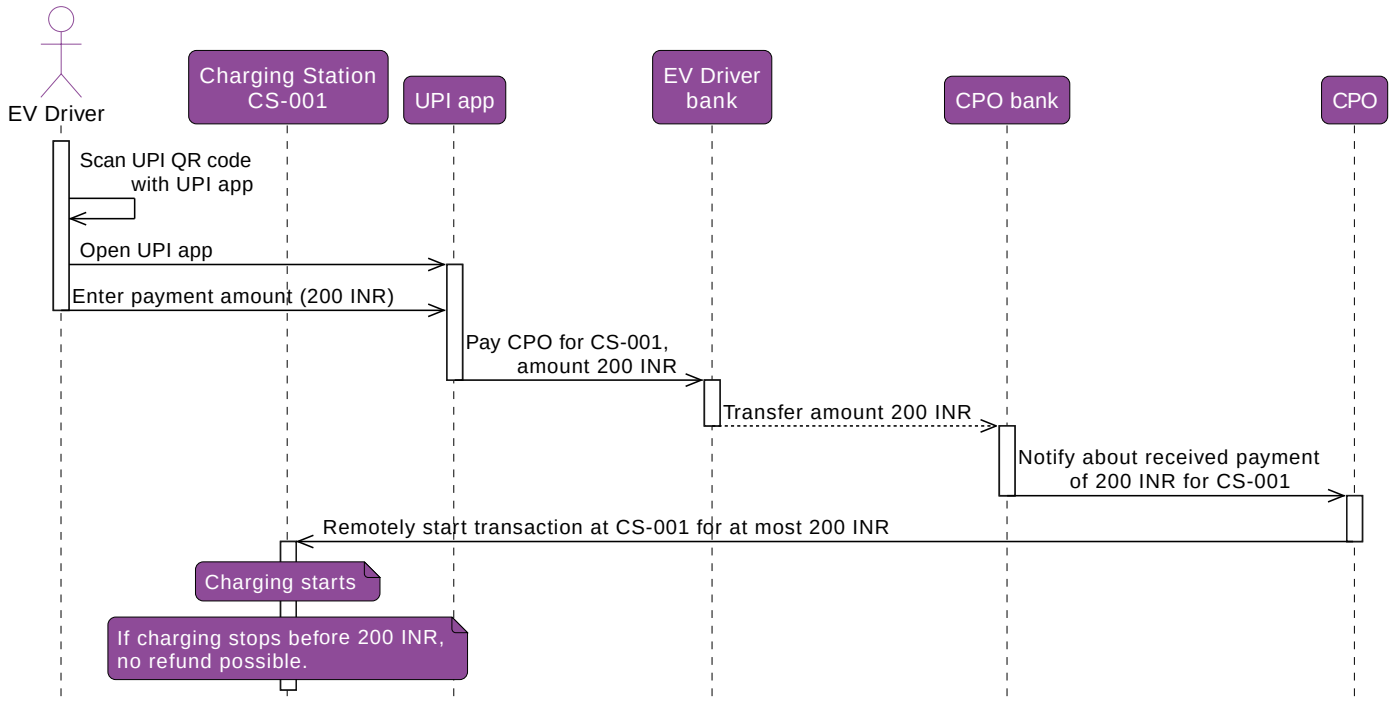

Figure 2. Steps for a pure UPI payment

2.1.2. Webpage-redirect QR code

Some operators take a different approach. Instead of referencing the charger directly in the UPI QR code, the UPI QR contains a `url` field with an HTTPS URL pointing to the operator's own server. This URL carries the charger identity in its query string, for example:

```
https://cpo.example.com/pay?charger_id=EVSE1234-1
```

When the user scans the QR code with any UPI payment app, their phone opens the CPO's webpage. The page can display the charger and session details, allow the user to select or confirm the payment amount, and present multiple payment options, including UPI. The UPI payment link is then generated dynamically on the webpage and passed to the UPI app once the user chooses that payment method.

This method gives the CPO more control over the payment flow and allows additional functionality (such as showing terms or offering alternative payment types). It allows the process to support easy refunds or adjustments via the API of the payment service provider (PSP). The drawback is that it requires deeper technical integration and merchant onboarding with a PSP, which adds complexity compared to the static approach.

The UPI payment server is not part of the customer's bank or the CPO backend. It is operated by the PSP or acquiring bank and serves as the intermediary that accepts the UPI payment, verifies success with NPCI, and notifies the CPO backend via an API or webhook. It also enables refund processing and transaction reporting.

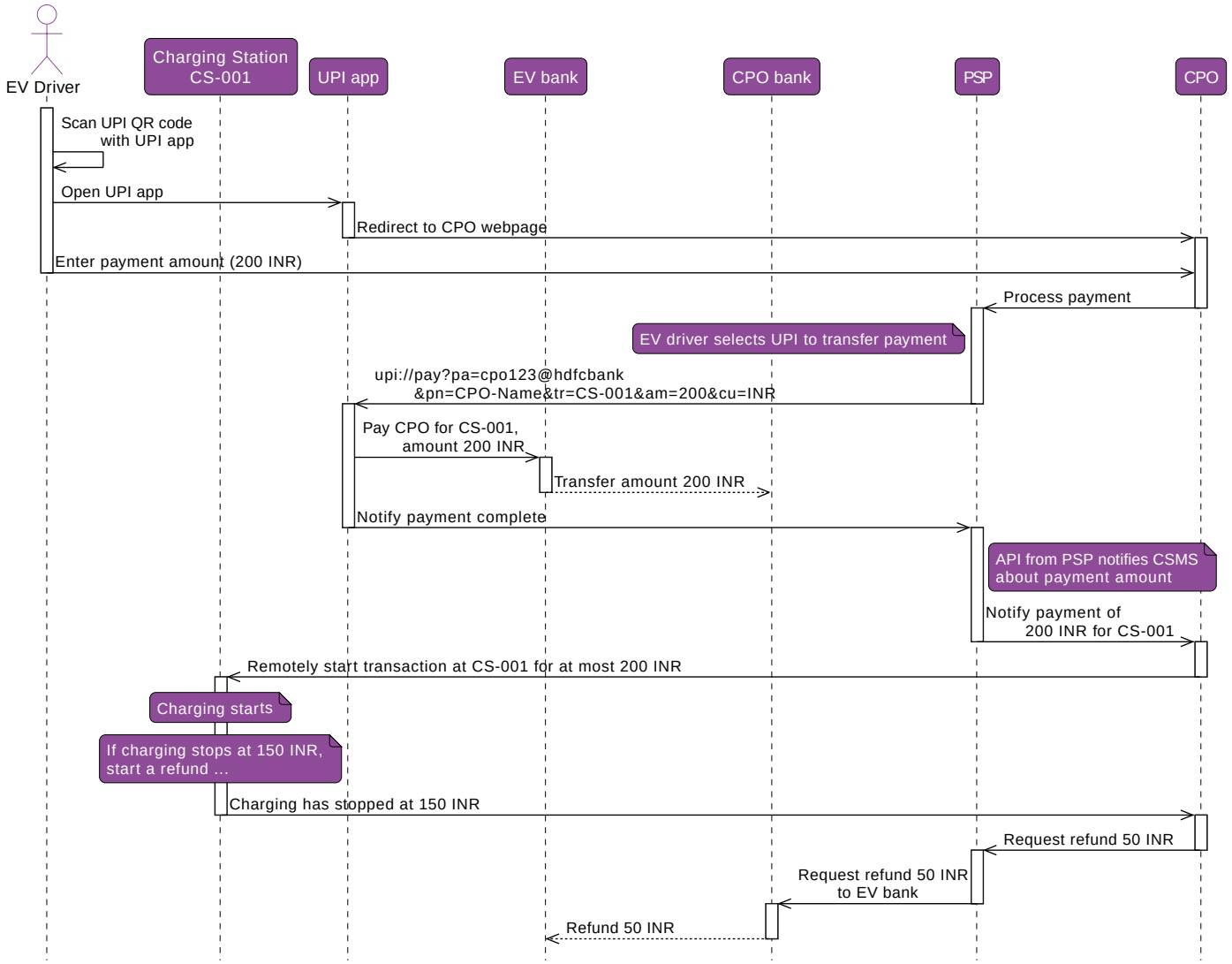

Figure 3. Steps for a webpage-redirect payment

2.2. When to use which model

The static QR method is well-suited for deployments where low cost and quick setup are more important than automation, such as a small public charger in a standalone location. Dynamic PSP integration, on the other hand, is better suited for networked, OCPP-compliant charging networks where seamless payment-to-charging automation and minimal operational overhead are key goals.

2.3. The future: postpaid and Autopay

While most UPI charging payments in India are prepaid today, the introduction of UPI Autopay could make postpaid charging feasible. This would work similarly to credit card delayed capture, where a driver can plug in first and pay the exact amount at the end of the session without the need for refunds or prepayment estimates.

3. How OCPP supports prepaid payments

Ad hoc payments are payments that are not tied to a subscription model, i.e. there is no relation between the CPO or charge card provider and the customer. The most well-known method of ad hoc payment is direct payment via a credit or debit card using a payment terminal. UPI payments are both ad hoc (not tied to a subscription) and prepaid (an amount is paid prior to charging).

A ad hoc session via credit or debit card is in fact similar to a prepaid charging session, even though the exact amount is debited at the end of the session. When paying with a credit or debit card, the CPO requests a reservation on the customer's bank account for a certain amount of money before starting the charging session, to be sure that enough funds are available to pay for the cost. When the cost of a charging session is about to exceed this amount, the charging session will be ended, unless it is possible to increase the reservation amount.

In OCPP version 1.6 and 2.0.1 there was no explicit support for prepaid and ad hoc payments. Although many implementations exist that support payment terminals, these are all bespoke solutions. As of version 2.1 OCPP offers explicit support for both prepaid and ad hoc payments using integrated payment terminals, payment kiosks or dynamic QR codes.

The scenarios for prepaid mobile payments that are described in this paper, evolve around the following steps:

1. the user paying a prepaid amount to the operator,
2. the operator backend (CSMS) remotely starting a charging session for the user,
3. the CSMS stopping the charging session when the prepaid amount runs out.

This process relies on two OCPP concepts: remotely starting a transaction, and limiting the transaction to a maximum energy or cost.

3.1. Step 1: Paying a prepaid amount

In case of UPI mobile payment the user pays a prepaid amount directly via the UPI payment app (Pure UPI static QR code) or via the CPO webpage (Webpage-redirect QR code). This takes place outside the charger and is out-of-scope of OCPP.

To prevent the risk of scanning a fake QR code that has been pasted over the operator's original code and redirects to a malicious website, the url field in the UPI QR code can use the added security of dynamic QR codes on a display with the time-based one-time password protection of OCPP 2.1 (see use case C25). This ensures that the scanned URL is only valid for a short period (e.g. 30 seconds). This option does require that the charger is equipped with a display that is able to show a QR code.

3.2. Step 2: Remotely starting a transaction

The backend (CSMS) can remotely start a transaction on a charging station by issuing the `RemoteStartTransaction` command for a specific EVSE (connector) of the charging station. After acknowledging the message the charging station will start a transaction on the specified EVSE and send a `StartTransaction` message when the transaction has started.

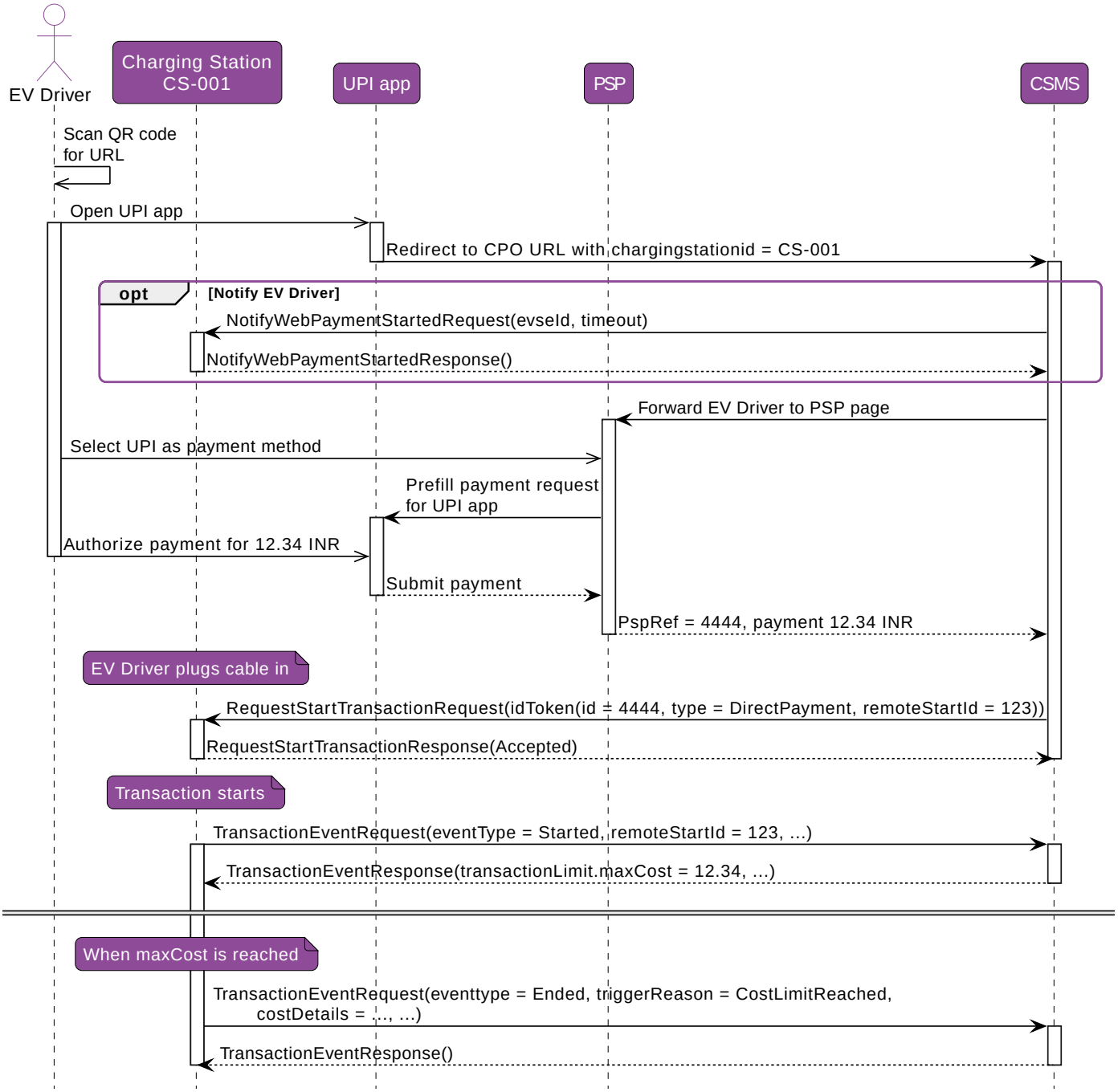

Figure 4. OCPP prepaid transactions

3.3. Step 3: Limiting transaction cost or energy

Prior to OCPP 2.1 the only way to stop a transaction when reaching a certain energy or cost limit, was by having CSMS continously check the meter values coming from the charging station, and stopping as soon as energy amount or calculated cost reached (or passed) the limit. This was never exact, because meter values are only sent at certain intervals.

Since OCPP 2.1 there is a new "transaction limit" feature. This adds a *transactionLimit* value to transactions by means of which the transaction can be limited in time, energy, cost and even state of charge of the vehicle. When CSMS receives a `StartTransaction` message, and it wishes limit the cost or energy of the transaction, then CSMS will add a *transactionLimit* to the response message that acknowledges the `StartTransaction` message.

In order to avoid exceeding the prepaid amount, the operator can choose to use a *transactionLimit* with either a

cost (*maxCost*) or an energy (*maxEnergy*) limit. For a cost limit the charger must be able to locally calculate the transaction cost. Not all chargers are capable of that. It is therefore, in many cases more practical to convert the prepaid cost to the amount of energy that equates to this amount.

If the energy price is 10 INR per kWh, then a prepaid amount of 200 INR equates to a maximum energy of 20 kWh. The CSMS will then set a *transactionLimit* with *maxEnergy* = 20 kWh. As soon as the consumed energy amount reaches 20 kWh, the charger will stop the energy transfer. It does not require monitoring by CSMS anymore.

OCPP 1.6 & 2.0.1

The older OCPP releases, 1.6 and 2.0.1, do not offer full support for ad hoc and prepaid payments. However, these releases do include a customization mechanism, which one can use to create a bespoke solution to support the mechanism described in this paper.

In OCPP 2.0.1 one can add a so-called *customData* extension to the TransactionEvent message to add the *transactionLimit* from OCPP 2.1 to the message. In OCPP 1.6 a possible solution might be to define a DataTransfer message to communicate the *transactionLimit* to the charger.

The OCA white paper "Customizing OCPP Implementations" (https://openchargealliance.org/ocpp-info-whitepapers/customizing-ocpp-implementations/) describes in detail how a customization can be added to your OCPP implementation.