



OPEN
CHARGE
POINT
INTERFACE

Tariff simplification in the EV charging market using OCPI v2.3

Issues and solutions for better exchange and understanding of EV charging tariffs

<https://github.com/ocpi> & <https://evroaming.org>

document version 1.0, 2026-07-09

Table of Contents

Preface	3
Updates/next versions	3
OCPI v2.3	3
Reducing complexity in the OCPI v2.3.0 tariff module: what is "complex"?	3
Functional vs technical guidelines	4
Ad Hoc tariffs	4
Part 1: Functional guidelines	5
1. Share tariffs between CPO and MSP via OCPI, before they become applicable for the EV driver	5
2. Keep tariffs understandable	5
Part 2: Technical guidelines	7
1. Always use step_size 1	7
2. Use restrictions only when the price changes	7
3. Usage of non-restricted elements	9
4. If there are multiple elements, there should only be a single component per element	9
5. Restrictions that are always or never applicable (redundant restrictions)	10
6. Usage of 0 (zero) in restrictions	10
7. Avoid creating duplicate tariff objects for the same pricing and validity dates	11
8. Avoid changing existing tariffs	13
9. Set end_date_time when a tariff replaces another one	14
10. Do not use restrictions that are hard to track or influence	14
11. Do not use conflicting tariff elements	14
12. Duration gaps	15
13. Wrong setting for "first X minutes free"	15
14. Don't use wrapping time restrictions or end_time 23:59	15
15. Push tariffs before using them	16
16. Don't use PARKING_TIME if there is no difference with the TIME price	16
17. Consistent use of validity dates	16

Copyright © 2014 – 2026 EVRoaming Foundation. All rights reserved.

This document is made available under the *Creative Commons Attribution- NoDerivatives 4.0 International Public License* (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

EVRoaming Foundation



OCPI is developed and managed by the EVRoaming Foundation. The EVRoaming Foundation is a contributor based organisation. Everyone can join the EVRoaming Foundation via <https://www.evroaming.org>

The EVRoaming Foundation strives to keep OCPI as free from IPR as possible. If you want to contribute by adding new functionality/features, you are required to send us the signed Contributor Agreement (CA) document before contributing. To get the CA, ask for it by send an e-mail to: info@evroaming.org.

Version History

Version	Date	Authors	Description
1.0	2026-07-09	OCPI Development Workgroup	Publication of the tariff simplification guide

Preface

Charge tariffs for EV drivers are still often experienced as complex and not sufficiently transparent, which frequently results in tariffs that are hard to understand. EV driver tariffs can either be Ad Hoc tariffs, where you pay directly at the charge station without any registration or subscription, or they can be provided via an MSP, which often calculates the EV driver tariff based on its buying tariff from the CPO. In these situations, the complexity of EV driver tariffs can be caused by complex tariffs between the CPO and the MSP.

Tariffs between CPOs and MSPs are exchanged via the OCPI Tariff module. This tariff module is complex and can be implemented in many different ways.

This document is the result of experiences from CPOs, MSPs and other parties in the market with the difficulties they encounter in the usage and implementation of the OCPI tariff module. It aims to:

- simplify the implementations
- prevent errors in implementations
- unify the way the tariff module is implemented.

If followed by all parties, the tariff exchange between CPOs and MSPs will be less complex and contain fewer errors.

Connected to this document is a set of Test Cases and a Tariff simplification test tool. The EVRoaming Foundation takes the content of this document into account to improve OCPI and its documentation.

This is not expected to solve all the issues that EV drivers experience with tariffs, but it should at least reduce them.

This document and the connected test tool are not the end; they are the beginning.

Input, remarks and comments are very welcome and can be sent to operations@evroaming.org

Updates/next versions

Based on new or changing developments, insights, experiences, research and feedback on this document, yearly updates will be made by the EVRoaming Foundation, together with its contributors.

OCPI v2.3

This document relates to the tariff module of OCPI v2.3.0, although most recommendations and guidelines can also be used in combination with v2.2.1. This document will be updated for future versions if needed.

Reducing complexity in the OCPI v2.3.0 tariff module: what is "complex"?

This document aims to reduce complexity in the implementation and usage of the OCPI v2.3.0 tariff module. Reducing complexity means finding situations where the same or similar results can be achieved with simpler implementations. Although the focus is on the tariff module of v2.3.0, the guidelines can also be used for earlier versions.

Functional vs technical guidelines

Functional guidelines are recommendations related to what the end user sees. The technical guidelines are about improving the way tariffs are represented and communicated without changing the "behaviour" of the tariff.

Ad Hoc tariffs

For CPOs active in European Union countries, ad hoc access with direct payment must be made available for their publicly accessible charge stations. This means that, for a publicly accessible charge station, a tariff with the [AD_HOC_PAYMENT] tariff type must be added. That way it can also be shared with the National Access Points (NAPs). These are EU member state data systems that contain information — including ad hoc tariffs — on all publicly accessible charging stations.

Part 1: Functional guidelines

1. Share tariffs between CPO and MSP via OCPI, before they become applicable for the EV driver

A large part of all tariff issues is related to the following:

- Tariffs between CPOs and MSPs are only shared via email or a website
- Tariffs are shared at the very last moment before they become applicable
- Ad hoc tariffs are not added via the Ad Hoc tariff type

All of these create big issues. Either the MSP cannot use the tariffs to calculate its EV driver tariffs and inform EV drivers in time, or — when ad hoc tariffs are incomplete — the EU National Access Points cannot share them with other interested parties and, ultimately, with the EV driver.

CPOs must share tariffs via OCPI. Failing to do so means that the MSP cannot create realistic EV driver tariffs.

Tariff changes must be shared BEFORE they become applicable. That way, the MSP has time to use them to calculate its EV driver tariffs and can inform the EV driver in time.

Limit the number of tariff changes. Frequent changes make it hard for the MSP to keep track of them and to keep its EV driver tariffs accurate, and they are confusing for the EV driver. If the goal is to change EV driver charging behaviour, that will not be achieved by frequently changing tariffs.

Price changes based on certain conditions (including during sessions, e.g. time-of-use pricing) should be part of the tariff. They do not require tariff changes.

NOTE

These periods are based on information from the market about the latest moments at which parties can know their own tariffs and determine the tariffs for their partners (MSPs or EV drivers).

The goal of sharing tariffs via OCPI is to avoid unexpected tariff increases.

2. Keep tariffs understandable

When you translate a tariff into human-readable language, it can become clear that you have reached the limit of what MSPs and their customers can be expected to comprehend. For example, consider the following tariff structure:

"When you charge less than 5 kWh, and it is before 8:00, and you have been charging for less than 2 hours, then the price per kWh is 61 cents.

Else, if you charge less than 5 kWh, then it is 71 cents per kWh.

In all other cases it is 55 cents per kWh.

In addition, when you charge less than 10 kWh there is a flat fee of 1.00 EUR.

In all other cases there is a flat fee of 50 cents."

Does something like this sound too complex or not?

If the answer is yes, then perhaps three restrictions per element is too many — and perhaps three elements per dimension is too many as well.

The use of dynamic tariffs, which follow fluctuating day-ahead prices, can be useful for changing charging behaviour. In this case too, you should translate the tariff into human-readable language and check whether it is still clear and understandable.

Part 2: Technical guidelines

Guidelines for a standardized way in which CPOs and MSPs can handle tariffs on a technical level, based on the Tariff module from OCPI v2.2.1 and 2.3.0.

The guidelines are based on "Tariff module common mistakes", as experienced by market participants.

Where possible, concrete 'good way' and 'wrong way' examples are given. The good ones are in green boxes; the wrong ones are in red boxes.

1. Always use step_size 1

A major source of confusion when implementing OCPI, and when data is shared between parties, is the usage and interpretation of [step_size]. It regularly causes confusion. [step_size] is only relevant for rounding at the end of transactions, for both dimensions (e.g. time or kWh) and price. This task can just as well be done by the system itself and does not need to be handled by OCPI. It is advised that any rounding of a given value is mentioned and agreed as part of the roaming agreement, so that the other party can take it into account.

From OCPI v3.0 [step_size] is also removed from the specifications. As it is a mandatory field in earlier versions, it is advised to keep the value always at "1".

Step sizes that are misinterpreted by the MSP when received from the CPO can result in wrong or confusing messages to the EV driver. It is much easier for parties to share the rounding they use via an agreement than to rely on the confusing step sizes.

2. Use restrictions only when the price changes

Restrictions are meant to indicate a situation or condition where the price changes. It is still very common to see restrictions used when nothing is changing. For example, tariffs that are shared by the CPO and duplicated for every single day, requiring the MSP to simplify them on its end to show a simple price to the user.

If the MSP were to take this over via an automated system for the end user tariff, the end user would see seven prices, one for every day of the week, that are all exactly the same.

OCPI gives us many ways to represent the same price. Let's take an example:

Let's say we want charging costs to be 0.50€/kWh.

Correct way: we can express this by simply defining a single segment containing a single energy component:

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}]
  }]
}
```

Incorrect way: we can also express it by defining seven segments, each with a day_of_week restriction and containing a single energy component:

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"day_of_week": [ "MONDAY" ]}
  },
  ...,
  {
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"day_of_week": [ "SUNDAY" ]}
  }
]
```

Incorrect way: or we can use multiple segments with the start_time / end_time restrictions:

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"start_time": "00:00", "end_time": "12:00"}
  },
  {
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"start_time": "12:00", "end_time": "00:00"}
  }
]
```

There are many more ways to represent the same price and they all come from abusing the segment's restrictions.

Most platforms are able to understand and process all these different implementations, but what do we show to the EV drivers?

1. 0.5€/kWh
2. 0.5€/kWh on Monday, 0.5€/kWh on Tuesday, ..., 0.5€/kWh on Sunday
3. 0.5€/kWh between 00:00 and 12:00, 0.5€/kWh between 12:00 and 00:00

They all mean the same thing: number one makes sense, the others probably don't.

A way to solve this problem is to use restrictions only when they are needed. In the examples above they are not needed because the price doesn't change from one segment to the other.

Example of a correct usage of restrictions according to the proposal

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"start_time": "00:00", "end_time": "12:00"}
  },
  {
    "price_components": [{"type": "ENERGY", "price": 0.60, "step_size": 1}],
    "restrictions": {"start_time": "12:00", "end_time": "00:00"}
  }
]
```

Example of an incorrect usage of restrictions according to the proposal

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"start_time": "00:00", "end_time": "12:00"}
  },
  {
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}],
    "restrictions": {"start_time": "12:00", "end_time": "00:00"}
  }]
}
```

The tariff above should be represented this way instead

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}]
  }]
}
```

3. Usage of non-restricted elements

Do not use a non-restricted element before a restricted element. The restricted element will never match in that case.

Example of a non-restricted element before a restricted element:

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.30, "step_size": 1}]
  },
  {
    "restrictions": {"end_time": "07:00", "start_time": "00:00"},
    "price_components": [{"type": "ENERGY", "price": 0.28, "step_size": 1}]
  }]
}
```

The example above should either switch the two elements or remove the second element, depending on the intention of the CPO.

4. If there are multiple elements, there should only be a single component per element

Since OCPI states that every "type" must be evaluated independently, only a single component per element should be used. This is also the reason why in OCPI 3.0 the different types (aka dimensions) have been completely split. Multiple components per element falsely suggest that all these components are always applied at the same time, while in practice that's not always the case.

5. Restrictions that are always or never applicable (redundant restrictions)

There are restrictions that are always applicable, e.g. all days of the week, or when there is no limit on the duration of a charging session. In these situations, the following options are seen:

1. Every element is shown e.g.: `day_of_week: [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY]`
 - a. This is redundant information; Guideline 2 explains why it should not be used this way.
2. The restriction is empty e.g.: `day_of_week: []`
 - a. This actually means that the element is valid on none of the weekdays (i.e. never). The CPO probably intends to express that there is no restriction on `day_of_week`.
3. The restriction is applicable for the full day: `start_time [00:00]` and `end_time [00:00]`
 - a. This means the whole day, which makes it irrelevant to add.
4. The restriction is `[0]` (zero): e.g. `min_duration: [0]`. See also the next guideline.
 - a. The correct way would be to remove this restriction, as it is always applicable. There is no other way to express this. If something applies on all days of the week, do not create a restriction for it.

6. Usage of 0 (zero) in restrictions

The OCPI specification gives no special meaning to the number 0 in restrictions.

Implementers should not assume that 0 (or 00:00 for time restrictions and [] for `day_of_week`) is a sentinel value meaning that a restriction field is not present at all. Some implementers have made such an assumption, resulting in tariff restrictions like `max_duration: [0]`. The operator probably means to indicate that there is no limit on the duration. However, in OCPI this restriction means the price component is never active.

An example of this is:

```
{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.7025, "step_size": 1}],
    "restrictions": {
      "start_date": "2025-10-22",
      "max_kwh": 0,
      "max_power": 150,
      "max_duration": 0,
      "end_time": "00:00",
      "day_of_week": []
    }
  }]
}
```

The MSP interprets this as: energy has a price, but as soon as you start consuming it, it is free.

The correct way would be:

```
{
  "id": "my-tariff",
```

```

"elements": [{
  "price_components": [{"type": "ENERGY", "price": 0.7025, "step_size": 1}],
  "restrictions": {
    "start_date": "2025-10-22",
    "max_power": 150
  }
}]
}

```

7. Avoid creating duplicate tariff objects for the same pricing and validity dates

MSPs receive and need to manage an increasingly large number of tariffs; as discussed in previous meetings, some of them already need to handle more than 100,000 tariffs. Most of those tariffs are just duplicates, but because they have different ids they still need to be managed.

A new tariff with the same elements but a different start_date_time should result in a new tariff instance being created. An existing tariff instance (with a historical start_date_time) can be applied to newly commissioned charging stations. This avoids unnecessary duplication of tariffs which follow the same pricing model.

This is not only a waste from a data-transfer point of view, but it also becomes a synchronisation problem.

A tariff needs to reach the MSP before it is attached to a connector; in OCPI terms, the tariff PUSH needs to happen before the connector PATCH:

PUT /tariffs/AA/BBB/my-tariff

```

{
  "id": "my-tariff",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.50, "step_size": 1}]
  }]
}

```

PATCH /locations/AA/BBB/my-location/my-evse/my-connector

```

{
  "tariff_id": "my-tariff"
}

```

This way, the MSP can make sure it shows the right price to its customers.

If a new tariff is created every time a price is changed, the synchronisation above is required.

Unfortunately, systems are not perfect, so any of the following can happen:

- A tariff might be pushed (way) after it is attached to the connector via a PATCH
- A tariff might not be pushed at all
- A tariff might not reach the EMSP (because the EMSP system is down or because the EMSP is rejecting it)

What should happen when a tariff is not delivered in time? This is perhaps a question for another proposal, but what can we do to reduce this risk?

In a perfect world, the CPO knows all the tariffs it will ever want to use and pushes them to the EMSP (or the EMSP pulls them) at handshake time. After that, the tariff module is not really needed anymore; the CPO just needs to send connector PATCHes to change tariffId using one of the ids it already communicated to the EMSP.

But this perfect world doesn't exist. Is there still something we can do? Yes! Reduce the need to push new tariffs (and hence the risk of having non-synchronised tariffs).

Example: Let's say a CPO has two tariffs in total:

```
{
  "id": "30-cent-kwh",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.30, "step_size": 1}]
  }]
}
```

```
{
  "id": "60-cent-kwh",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.60, "step_size": 1}]
  }]
}
```

The MSP will first pull the tariffs after handshake and then pull locations. Each connector has either tariffId=30-cent-kwh or tariffId=60-cent-kwh.

After some months, the CPO wants to change its prices:

- 30-cent-kwh should become 60-cent-kwh
- 60-cent-kwh should become 90-cent-kwh

Now, the MSP already has 60-cent-kwh, so there is no need to push it again, as there is no need to create a new tariff with the same content but a different id.

To change the price from 30-cent-kwh to 60-cent-kwh, all the CPO needs to do is:

PATCH /locations/AA/BBB/...

```
{
  "tariff_id": "60-cent-kwh"
}
```

The CPO does this for each connector that previously had tariffId=30-cent-kwh.

To change the price from 60-cent-kwh to 90-cent-kwh, a synchronisation is required, because the MSP does not know yet that 90-cent-kwh exists. In this case the CPO should:

PUT /tariffs/AA/BBB/90-cent-kwh

```
{
  "id": "90-cent-kwh",
  "elements": [{
    "price_components": [{"type": "ENERGY", "price": 0.90, "step_size": 1}]
  }]
}
```

```
}
```

Then, for each connector that previously had tariffId=60-cent-kwh:

PATCH /locations/AA/BBB/...

```
{
  "tariff_id": "90-cent-kwh"
}
```

If the CPO now wanted to revert the price change back to 30-cent-kwh and 60-cent-kwh it wouldn't need the tariff module anymore, because the MSP already has those two tariffs:

PATCH /locations/AA/BBB/...

```
{
  "tariff_id": "30-cent-kwh"
}
```

The CPO does this for each connector that previously had tariffId=60-cent-kwh.

PATCH /locations/AA/BBB/...

```
{
  "tariff_id": "60-cent-kwh"
}
```

The CPO does this for each connector that previously had tariffId=90-cent-kwh.

Bottom line: the CPO should send as many tariffs as it has different prices. This means:

- Only one tariff (and tariffId) saying 30 cent per kWh
- Only one tariff (and tariffId) saying 60 cent per kWh
- ...

NOTE

In these examples, it is assumed the end_date_time of the tariff object is still unset or in the future. If not, a new tariff object should be created, with a new id and applied to the relevant connectors.

8. Avoid changing existing tariffs

Never change a tariff which has already become active or has been active in the past. In OCPI 3.0 this way of working became mandatory. Updating a tariff makes it very unclear when the new tariff becomes valid. Tariffs should be published well in advance (so every MSP receives them well before the tariff becomes active) and then all the related connectors should be updated (also well in advance) to have two tariff IDs: one for the current tariff and one for the future tariff. This applies to OCPI 2.2.x and OCPI 2.3.

OCPI's tariff module allows updating/deleting already communicated prices via the PATCH/DELETE endpoints.

This behaviour introduces a number of challenges:

- Should the new tariff be valid from now? What about charging sessions that were already ongoing when the tariff update came?
- How can the tariff updates be traced? The id is always the same but the content is different.
- What does it mean to DELETE a tariff?

In the spirit of keeping things simple, the CPO could communicate tariff changes as shown in Proposal 2:

1. Create a new tariff when a new one is needed (i.e. a tariff whose contents were never communicated before).
2. PATCH the relevant connectors setting the new tariffId.

This way every tariff change is traceable by looking at the connector PATCHes and each tariff is immutable, meaning a tariff with id 90-cent-kwh always means the same thing.

It will also render tariff PATCH and DELETE useless, simplifying the tariff module implementations.

This also implies that one should use the `start_date_time` and `end_date_time` at the top level of a tariff to define the tariff validity. The `start_date` and `end_date` restrictions should NOT be used for this purpose. The `start_date` and `end_date` restrictions can be used for special rates on special days, like national holidays, as-per Guideline 17.

9. Set `end_date_time` when a tariff replaces another one

Guideline 8 mentions that a communicated tariff should not be changed. However, if a new tariff is published that will replace the current one, the `end_date_time` should be set on the old tariff. And if the old tariff is no longer in use on any connector (and is not intended to be used in the future), its `end_date_time` should be set as well.

10. Do not use restrictions that are hard to track or influence

There are a few restrictions that can be used in tariffs, but are hard to track or influence by the MSP and the EV driver. The main ones are:

- `min/max_current`: this requires people to understand how current, voltage and power relate.
- `min/max_power`: easier to understand than current, but often not something under the control of the MSP or customer.
- duration restrictions on energy components and kWh restrictions on time components. E.g. when you have consumed 20 kWh, the hourly price increases. For the EV driver it is difficult to track or influence when the 20 kWh is reached.

These restrictions should NOT be used, as they make the tariff complex without giving any influence over the charging activity, and the end user cannot track when they will take effect.

11. Do not use conflicting tariff elements

It happens that parties share the same component types, with the same restrictions, but with different prices. Although the OCPI specifications state that they must be processed in the order in which they appear, this forces MSPs to build defensive logic rather than rejecting invalid tariffs at ingestion. OCPI does not forbid the MSP from

rejecting these situations. Parties should be aware that conflicting segments should not be used and will result in rejection of the tariffs by the MSP.

12. Duration gaps

There are situations where tariffs have a duration gap, e.g. 0–3600 seconds = X, 3660+ = Y. Technically, this leaves unpriced time windows.

13. Wrong setting for “first X minutes free”

Some parties make the following mistakes:

- Misuse of PARKING + min_duration, where CPOs believe it means “first X minutes free”, but technically it means “charge if the session exceeds X”.
- Multiple flat fees (e.g. a start fee plus a long-stay fee) that encode business logic in ways that are unclear or contradictory.

The EVRoaming Foundation is working on an extension for v2.3.0 on how time-based fees that start after charging can be used.

14. Don’t use wrapping time restrictions or end_time 23:59

Don’t use end_time < start_time restrictions that wrap to the next day. This is confusing, especially when combined with a day_of_week restriction. Split the component into two components: one with a start_time and one with an end_time.

If a component should be active until the end of the day, use no end_time at all instead of end_time 23:59 (which leaves a gap between 23:59 and 00:00) or 00:00.

Incorrect:

```
{
  "id": "my-tariff",
  "elements": [{
    "restrictions": {
      "start_time": "22:00",
      "end_time": "8:00"
    },
    "price_components": [{"type": "TIME", "price": 0.50, "step_size": 1}]
  }]
}
```

Correct:

```
{
  "id": "my-tariff",
  "elements": [{
    "restrictions": {
      "start_time": "22:00"
    },
    "price_components": [{"type": "TIME", "price": 0.50, "step_size": 1}]
  }]
}
```

```
},
{
  "restrictions": {
    "end_time": "08:00"
  },
  "price_components": [{"type": "TIME", "price": 0.50, "step_size": 1}]
}]
}
```

15. Push tariffs before using them

Don't associate a tariff id with a connector before pushing a tariff with this tariff id.

16. Don't use PARKING_TIME if there is no difference with the TIME price

If a tariff has an hourly price for the complete session duration (time plugged in), use the TIME component. Only use PARKING_TIME if the price when not charging is different from the price when charging. Don't create a duplicate PARKING_TIME component for every TIME component.

17. Consistent use of validity dates

Use `start_date_time` and `end_date_time` on the object to specify the time window a tariff is valid for, i.e. when it was introduced and later decommissioned.

`start_date` and `end_date` on the restrictions of individual tariff elements are expected to be used only for short-term, exceptional pricing changes, e.g. for specific days that can be predicted at tariff creation time, such as public holidays.