

Smart Fuel Pass

OCPI Integration Guide for CPO Partners

How to connect your charging network to Smart Fuel Pass

Document version

1.2 · 05/2026 · Smart Fuel Pass

Intended for

Charge Point Operators (CPO) integrating with Smart Fuel Pass (eMSP)

Contact

techsupport@smartfuelpass.com

Table of contents

1	Introduction	4
1.1	Supported OCPI versions	4
1.2	Official OCPI documentation & resources	4
2	Session model in Smart Fuel Pass	6
2.1	Two types of sessions	6
2.1.1	Ad-hoc session (SFP internal)	6
2.1.2	OCPI Session (CPO side)	6
2.2	Relationship between sessions	6
2.3	OCPP to OCPI State Mapping	7
2.4	Session flow — overview	8
3	What you need to prepare	9
3.1	Information to send us	9
3.2	Technical requirements on your side	9
4	OCPI modules — your role as CPO	10
5	Charging session behaviour and start logic	11
5.1	Charging start is defined by Session, not Command	11
5.2	Session creation requirement and timeout	11
5.3	Timeout handling and failed start	11
5.4	GET /sessions as fallback validation	11
5.5	Session alone is not sufficient — charging validation	11
5.6	DC handshake failure handling — automatic retry	12
5.7	Connector state has priority over Session state	12
5.8	Session continuity requirement	12
5.9	Charging start trigger strategies	12
5.9.1	Immediate START	12
5.9.2	Wait-for-PREPARING (connector-driven start)	12
5.9.3	Purpose of PREPARING-based start	13
5.9.4	Activation	13
5.10	STOP_SESSION behaviour	13
5.11	Meter values requirements	13
5.12	Financial flow independence	13
6	What Smart Fuel Pass provides to you	14
7	Integration process — step by step	15
7.1	Stage 0 — Pre-integration (your action required)	15
7.2	Stage 1 — SFP infrastructure setup (no action from you)	15
7.3	Stage 2 — OCPI handshake	15
7.3.1	Method A — Automatic handshake (recommended for new integrations)	15
7.3.2	Method B — Manual credential setup (for re-registration or migration)	16
7.4	Stage 3 — Functional testing	16
7.5	Stage 4 — Go-live & sign-off	16
8	Common issues & how to resolve them	17
9	Quick reference checklist	18

10	Contact & support	19
----	-------------------------	----

1 Introduction

Smart Fuel Pass (SFP) is an E-Mobility Service Provider (eMSP) operating in Slovakia and across Central Europe. We enable EV drivers to access charging networks seamlessly through our app and card — regardless of the charging operator.

This document is your complete guide for connecting your charging infrastructure to the Smart Fuel Pass platform via the OCPI protocol. It covers everything you need to prepare, what we provide, and the step-by-step process to go live.

Roles in this integration	
Smart Fuel Pass → eMSP (E-Mobility Service Provider)	— we manage EV driver accounts, authorisation, and billing.
Your company → CPO (Charge Point Operator)	— you operate charging stations and handle the charging sessions.
OCPI → the open protocol we use to exchange data between our systems.	

1.1 Supported OCPI versions

Version	Status	Notes
OCPI 2.1.1	Supported	Single endpoint per module. Suitable for most integrations.
OCPI 2.2	Supported	Sender + Receiver endpoints per module.
OCPI 2.2.1	Supported — recommended	Most widely adopted version. Preferred for new integrations.
OCPI 2.3.0	Available on request	Includes AFIR-related modules (Booking, Payment Terminal). Not enabled by default – contact us if your integration requires it.

Please confirm your supported OCPI version with us before starting the technical setup.

Other OCPI versions
Please contact our Technical Support on techsupport@smartfuelpass.com for more information.

1.2 Official OCPI documentation & resources

The OCPI API itself is not the subject of this document, as it is a standardized interface defined by the EVRoaming Foundation. The implementation of individual OCPI endpoints, data models, and behaviours must follow the official specification (references are provided in the table below) and is the responsibility of each integration party.

This document does not aim to describe how to implement the OCPI standard, but rather how it is used within a specific eMSP–CPO integration scenario. Its purpose is to define the expected behaviour and event flows required to reliably initiate and stop charging sessions, as well as to ensure correct billing through a payment terminal or an e-commerce payment gateway on the eMSP side.

For detailed technical specifications, refer to the official OCPI documentation maintained by the EVRoaming Foundation.

Resource	URL
OCPI 2.1.1 specification	https://github.com/ocpi/ocpi/releases/tag/2.1.1
OCPI 2.2 specification	https://github.com/ocpi/ocpi/releases/tag/2.2

Resource	URL
OCPI 2.2.1 specification	https://github.com/ocpi/ocpi/releases/tag/v2.2.1-d2
OCPI 2.3.0 specification	https://github.com/ocpi/ocpi/releases/tag/v2.3.0
EVRoaming Foundation (official maintainer)	https://evroaming.org/ocpi/
OCPI GitHub repository	https://github.com/ocpi/ocpi

2 Session model in Smart Fuel Pass

Before diving into technical requirements, it is helpful to understand how Smart Fuel Pass models charging sessions internally. This context explains some of the behaviours described in section 5 (Charging session behaviour and start logic).

2.1 Two types of sessions

2.1.1 Ad-hoc session (SFP internal)

An ad-hoc session is created when a user initiates charging — for example via QR code, web link, or payment terminal.

This session:

- manages the full charging attempt lifecycle
- controls financial pre-authorisation
- handles retries and timeouts
- is visible to the end user

An ad-hoc session may contain zero, one, or multiple charging attempts.

2.1.2 OCPI Session (CPO side)

An OCPI Session represents an actual charging process on the CPO side.

- It is created only when charging starts on the charger.
- It is used for real-time updates, energy tracking, and billing (CDR).

2.2 Relationship between sessions

One ad-hoc session can produce	Zero OCPI sessions (if charging never starts), or multiple OCPI sessions (in case of retries).
Valid for billing	Only OCPI sessions that successfully start and deliver energy.

Important implication for CPO

Multiple START_SESSION commands may be sent within a single user interaction.

Not all resulting OCPI sessions are guaranteed to be valid or settled.

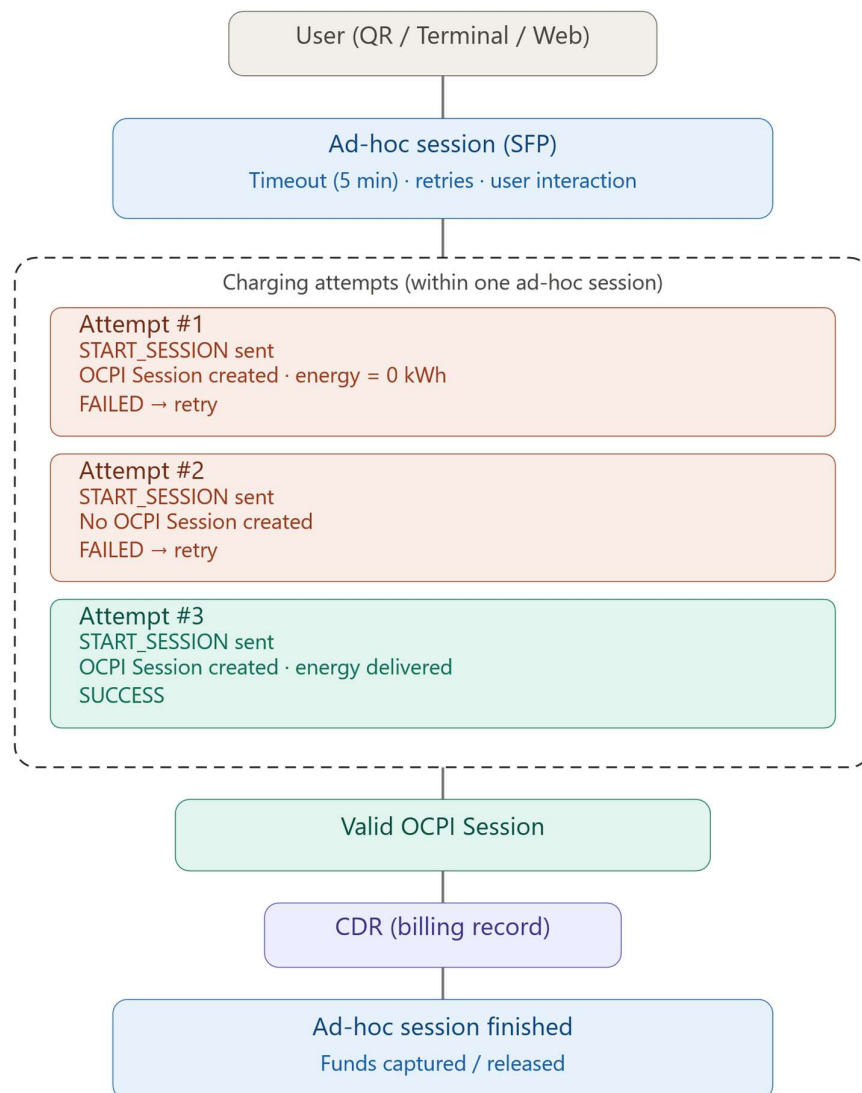
Only sessions that meet SFP validation criteria (session active + energy delivered) will be used for billing.

2.3 OCPP to OCPI State Mapping

The table describes the mapping of charging point states between the OCPP protocol (communication between the charging station and our backend) and the OCPI protocol (communication between our backend and eMSPs / roaming platforms). Since OCPP works with finer state granularity than OCPI, multiple OCPP states are consolidated into a single OCPI state. More information on section [5. Charging session behaviour and start logic](#).

OCPP state	OCPI state
Available	AVAILABLE
Preparing	BLOCKED
Charging	CHARGING
SuspendedEv	CHARGING
SuspendedEvse	CHARGING
Finishing	CHARGING
Faulted	OUTOFORDER
Reserved	RESERVED
Unavailable	OUTOFORDER
New	PLANNED
Removed	REMOVED
Unknown	UNKNOWN
Offline	UNKNOWN

2.4 Session flow — overview



3 What you need to prepare

Before we can start the integration, you need to prepare the following. This should not take more than a few hours on your side.

3.1 Information to send us

Please send the following to techsupport@smartfuelpass.com before we schedule the integration:

	Item	Format / Example	Status
☐	OCPI Versions URL Full URL to your OCPI versions endpoint	https://your-domain.com/ocpi/cpo/versions	
☐	Token A One-time initial token for OCPI handshake — UUID format, generated on your side	a1b2c3d4-e5f6-7890-abcd-ef1234567890	
☐	Party ID 3-character alphanumeric code (A–Z, 0–9) identifying your company in the OCPI network, as defined by eMI3 / IDACS	ION / GRN / ABA	
☐	Country Code ISO 3166-1 alpha-2 code of your country of operation	CZ / SK / AT / DE	
☐	OCPI version Which version you support: 2.1.1, 2.2, 2.2.1 or 2.3.0	—	
☐	Static IP address The outbound IP address (or CIDR range) of your OCPI server — needed for our firewall whitelist	1.2.3.4 or 1.2.3.0/24	
☐	Technical contact Name, email and phone of your technical contact person for the integration	—	

3.2 Technical requirements on your side

Your OCPI server must meet these requirements before we start testing:

	Requirement
☐	Your OCPI server is publicly reachable from the internet or from our specific IP addresses We need to be able to call your Versions endpoint during the handshake. We will provide our outbound IP addresses — add them to your firewall / WAF allowlist.
☐	HTTPS with TLS 1.2 or higher We do not support plain HTTP connections in production.
☐	Authorization token Your server must provide the Token A in the Authorization header: Authorization: Token {Token}
☐	After handshake: store our token and endpoints Your system must persist the SFP credentials received during the handshake for all subsequent calls.
☐	Implemented OCPI modules confirmed See section 4 for the full list of required modules and your expected role per module.

4 OCPI modules — your role as CPO

OCPI is modular. Each module has a defined direction of data flow. As a CPO, you are the Sender for location and session data, and the Receiver for authorisation tokens and remote commands.

The table below shows what you are expected to implement:

Module	Your role (CPO)	SFP role	What this means for you
Locations	Sender	Receiver	You push your charging station data (location, EVSE, connector status) to us. We display it to drivers in our app.
Sessions	Sender	Receiver	You send live session updates while a driver is charging. We use this to show real-time charging status.
CDRs	Sender	Receiver	You send the Charge Detail Record at the end of each session. This is the basis for billing the driver.
Tariffs	Sender	Receiver	OPTIONAL: You send your pricing information. We display it to drivers before they start charging. Depends on business model.
Tokens	Receiver	Sender	We push our authorisation tokens to you. You cache them and use them to authorise charging sessions locally.
Commands	Receiver	Sender	We send remote commands (START_SESSION, STOP_SESSION). You execute them and return an async result.

Commands — important note

Commands are asynchronous in OCPI. When we send a START_SESSION command, your server returns an immediate acknowledgement (CommandResponse), then sends the final result via a separate POST to our callback URL (CommandResult).

Your system must implement both the synchronous response and the asynchronous callback.

5 Charging session behaviour and start logic

This section describes how Smart Fuel Pass evaluates charging sessions and manages charging start behaviour beyond standard OCPI message exchange. The goal is to ensure a consistent user experience, reliable session tracking, and correct billing — even in cases of partial or inconsistent CPO implementations.

5.1 Charging start is defined by Session, not Command

When SFP sends a START_SESSION command:

- The initial CommandResponse (e.g. ACCEPTED) is treated only as an acknowledgement.
- Charging is considered successfully started only when:
 - a Session object is created on the CPO side with ACTIVE state, and
 - delivered to SFP via PUT /sessions or available via GET /sessions.

5.2 Session creation requirement and timeout

After a START_SESSION is accepted:

- The Session must be created and visible within a defined time window.
- Default timeout: 5 minutes (unless agreed otherwise during integration).

If the Session is not received within the timeout:

- SFP attempts to retrieve it via GET /sessions (must be supported by the CPO).

5.3 Timeout handling and failed start

If no Session is found within the timeout:

- The charging attempt is considered unsuccessful.
- SFP releases the financial pre-authorisation.
- SFP allows the user to retry charging.

If a Session appears after the timeout has already expired:

- It is considered invalid from the SFP perspective.
- It may not be eligible for settlement.

5.4 GET /sessions as fallback validation

If no Session update is received via OCPI push (PUT /sessions), SFP uses GET /sessions as a fallback validation mechanism.

Outcome	Behaviour
Session exists	Charging continues to be tracked normally.
Session does not exist	Funds are released.
GET not available / failing	Funds remain reserved and the issue is escalated to the CPO.

5.5 Session alone is not sufficient — charging validation

A Session being created does not guarantee successful charging. SFP additionally evaluates:

- connector status transitions,
- delivered energy (kWh),
- charging power (kW).

Charging is considered successful only when the session remains active and energy delivery is observed.

5.6 DC handshake failure handling — automatic retry

In DC charging scenarios, the following situation may occur:

- Session is created and connector enters CHARGING state,
- but no energy is delivered (0 kWh),
- and connector falls back to AVAILABLE or PREPARING.

In this case the attempt is considered unsuccessful and SFP automatically retries START_SESSION:

- within the active pre-authorisation window,
- until charging successfully starts or the timeout is reached.

5.7 Connector state has priority over Session state

SFP prioritises physical connector state over Session state. If the connector transitions to AVAILABLE or PREPARING while a Session is still marked as active:

- the Session is considered effectively ended or failed,
- SFP may release funds,
- a new charging attempt may be initiated.

This prevents blocking the user due to inconsistent or stale session data from the CPO.

5.8 Session continuity requirement

A valid charging session must remain active, continue delivering energy, and maintain a consistent connector state.

If the session remains active but no energy or updates are received, or if connector state contradicts session state, SFP may stop tracking the session and allow a new session to be started.

5.9 Charging start trigger strategies

SFP supports multiple strategies for triggering START_SESSION, depending on CPO capabilities. These strategies are configurable per CPO or per charger.

5.9.1 Immediate START

START_SESSION is sent immediately after financial pre-authorisation is confirmed.

5.9.2 Wait-for-PREPARING (connector-driven start)

In this mode:

- The user completes financial pre-authorisation.
- SFP waits for the connector state to become PREPARING (typically mapped from OCPP Preparing) or BLOCKED.
- Only after this state is observed does SFP send START_SESSION.

If the connector remains in AVAILABLE:

- START_SESSION is not sent.
- SFP waits until timeout, after which the attempt is considered unsuccessful and funds are released.

5.9.3 Purpose of PREPARING-based start

This mode is particularly useful in environments where session creation is delayed or inconsistent, or where the CPO does not reliably create Session objects immediately. By waiting for connector readiness, SFP ensures that charging is physically initiated and reduces the risk of failed or empty session scenarios.

5.9.4 Activation

Optional feature
The PREPARING-based start strategy is optional and enabled only if supported by the CPO and agreed during integration.
Unless explicitly agreed, SFP uses the immediate start model by default.

5.10 STOP_SESSION behaviour

STOP_SESSION is treated as a request, not a guaranteed stop. The final state of the session is determined by subsequent session updates or the final CDR.

5.11 Meter values requirements

Parameter	Detail
Required fields	Energy (kWh) and Power (kW) — mandatory for every session.
Optional fields	State of Charge / SoC — recommended for better user experience.
Expected update interval	Approximately every 60 seconds during active charging. If no sample is received within the expected interval, SFP attempts to retrieve session details via GET /sessions.
Data handling	Out-of-order or delayed values may be ignored. SFP relies on timestamp consistency.

5.12 Financial flow independence

Financial pre-authorisation is independent of connector state.

- Funds are reserved before charging starts.
- Funds are released if no session is created, if the session fails to start, or if the timeout is reached.

Connector states such as BLOCKED or PREPARING do not directly affect the financial logic — they influence the start trigger only.

6 What Smart Fuel Pass provides to you

Once you have sent us the information from section 3, we will share the following before the integration starts:

Item	Value
SFP OCPI Versions URL	https://ocpi-{yourcode}.smartfuelpass.com/ocpi/emsp/versions
Token B (SFP side)	UUID token — sent via secure channel (email or encrypted message). <i>After a successful handshake, both systems exchange permanent tokens automatically.</i>
Party ID	SFP
Country Code	SK
Business Name	Smart Fuel Pass
Website	www.smartfuelpass.com
SFP outbound IP addresses	For your firewall / WAF allowlist — sent before testing begins

7 Integration process — step by step

The integration follows five stages. Your active involvement is needed in stages 0, 2, 3, and 4.

#	Stage	Description	Who	Time
0	Pre-integration	Exchange of information and credentials via email. Agree on OCPI version and schedule.	Both	1–2 d
1	SFP infrastructure setup	We set up the dedicated OCPI endpoint for your integration on our side.	SFP	2–4 h
2	OCPI handshake	Automated or manual exchange of credentials. Both systems register each other.	Both	30–90 m
3	Functional testing	Locations sync, token authorisation, session flow, CDRs, commands.	Both	1–3 h
4	Go-live & sign-off	Production deployment and confirmation from both sides.	Both	

7.1 Stage 0 — Pre-integration (your action required)

1. Send us all items from section 3.1 to techsupport@smartfuelpass.com.
2. Confirm which OCPI version you support (2.1.1 / 2.2 / 2.2.1 / 2.3.0).
3. Agree on an integration window (date and time) with our technical team.
4. Ensure your OCPI server is running and publicly reachable at the agreed time.

7.2 Stage 1 — SFP infrastructure setup (no action from you)

We will set up a dedicated OCPI endpoint for your integration:

Parameter	Value
Your SFP endpoint	https://ocpi-{yourcode}.smartfuelpass.com
Party ID	SFP
Country Code	SK
OCPI version	As agreed

We will notify you when setup is complete and ready for handshake.

7.3 Stage 2 — OCPI handshake

The handshake is the technical registration step where our systems exchange credentials. There are two methods:

7.3.1 Method A — Automatic handshake (recommended for new integrations)

Our system initiates the handshake automatically using the credentials you provided (Versions URL + Token A). No manual configuration is needed on your side beyond having your server ready.

1	We configure your Versions URL and Token A on our side.
2	Our system contacts your Versions URL using Token A.
3	Both systems automatically exchange permanent tokens and endpoint URLs.
4	Registration is complete — both systems are connected.

7.3.2 Method B — Manual credential setup (for re-registration or migration)

Used when your system has already registered SFP before (e.g. after a re-deployment). We configure the credentials directly without a new handshake. We will handle this on our side. You do not need to reset your registration unless we ask you to.

☒ How to confirm the handshake succeeded
Send a test GET request to our Versions endpoint using your last used Token.
Expected response: HTTP 200 with a list of supported OCPI versions.
HTTP 401 means the token is not yet accepted — contact us to verify on techsupport@smartfuelpass.com.

7.4 Stage 3 — Functional testing

Once the handshake is complete, we run through the following tests together. Please have your technical contact available during this stage.

	Test	Who validates	Status
☐	Locations sync — your station data is visible in our system	Both	
☐	Session start — charging session starts and is tracked correctly	Both	
☐	CDR received — Charge Detail Record arrives after session ends (remote or local stop)	SFP	
☐	Remote command — START_SESSION or STOP_SESSION executes correctly	Both	
☐	Session timeout handling — session behaviour observed if charging does not start	Both	
☐	Connector state transitions — AVAILABLE → PREPARING → CHARGING flow verified	Both	
☐	Meter values — kWh and kW received within expected interval	SFP	

7.5 Stage 4 — Go-live & sign-off

After successful testing, we move to the production environment and both sides confirm the integration is live.

	Action	Who
☐	Production environment tested and confirmed	Both
☐	You confirm successful end-to-end test from your side	CPO
☐	Sign-off form completed by both parties	Both
☐	Integration documented and archived	SFP

8 Common issues & how to resolve them

Symptom	Likely cause	Resolution
HTTP 403 when calling SFP endpoints	Your IP is not in our firewall allowlist	Confirm your outbound IP with us — we will add it to Cloudflare WAF.
HTTP 403 when SFP calls your endpoints	Our IP is not allowlisted on your side	Add the SFP outbound IP addresses we provided to your firewall / WAF.
HTTP 401 on all requests after handshake	Token mismatch — wrong authorization Token	Verify the tokens exchanged during handshake. Contact us to re-check.
Handshake times out	Wrong Versions URL or Token A	Confirm the URL is publicly reachable (test with curl from external network) and the Token A matches exactly.
Locations not visible in SFP system	Locations not pushed or push failing	Check your outbound request logs. Confirm the SFP Locations endpoint URL is correct.
CDRs not received by SFP	CDR not sent, or sent to wrong endpoint	Verify the CDRs endpoint URL. CDR must be sent as POST immediately after session ends.
START_SESSION returns error	EVSE not available or command rejected	Return the correct CommandResponse code. Verify the EVSE ID matches your Location data.
Sessions not updating in real time	Session PATCH not implemented	You must send PATCH /sessions updates during an active charging session.
Charging starts but no energy reported	Meter values not sent or wrong interval	Ensure kWh and kW are sent approximately every 60 seconds. Check timestamps.
Session created but funds released early	No energy delivered within timeout	This is expected SFP behaviour (section 5). Ensure your charger delivers energy promptly after session creation.

9 Quick reference checklist

Use this as your personal pre-integration checklist before contacting us to schedule the handshake.

Item	
Provide to SFP	
☐	OCPI Versions URL Public URL of your versions endpoint
☐	Token A UUID, generated by you, sent via secure channel
☐	Party ID 3 uppercase letters
☐	Country Code ISO 3166-1 alpha-2
☐	OCPI version 2.1.1 / 2.2 / 2.2.1 / 2.3.0
☐	Static IP address Your outbound IP or CIDR range
☐	Technical contact Name, email, phone
Your server is ready	
☐	OCPI server publicly reachable over HTTPS
☐	Token A accepted in Authorization header
☐	Locations module implemented (you as Sender)
☐	Sessions module implemented (you as Sender, including PATCH with kWh/kW every ~60s)
☐	CDRs module implemented (you as Sender, POST after session end)
☐	Tokens module implemented (you as Receiver — cache SFP tokens)
☐	Commands module implemented (you as Receiver — START/STOP at minimum, async CommandResult callback)
☐	OPTIONAL: Tariffs module implemented
☐	SFP outbound IP addresses added to your firewall / WAF
Receive from SFP	
☐	SFP Versions URL received (<i>provided by SFP prior to handshake</i>)
☐	SFP Token B received (<i>provided by SFP prior to handshake</i>)
☐	SFP Party ID (SFP) and Country Code (SK) noted <i>provided by SFP prior to handshake</i>
☐	SFP outbound IP addresses received and allowlisted

10 Contact & support

Type	Contact
Technical integration	techsupport@smartfuelpass.com
Production incidents	techsupport@smartfuelpass.com
Website	www.smartfuelpass.com

For all integration-related questions, please include your Party ID and the OCPI version you are using in your message. This helps us route your request to the right person immediately.