

Detailed connector status for improved user feedback

Jens Andersson

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

0.1, 2026-11-02: Draft

Table of Contents

1. Requirement terminology . . . 3
2. Motivation . . . 3
3. Domain . . . 3
    3.1. Use cases . . . 3
        3.1.1. UC1. As a CPO Platform, I want to push connector-level status transitions to partner Platforms in order to improve real-time interoperability . . . 3
        3.1.2. UC2. As an eMSP or app provider, I want to identify connectors that were just plugged in but have no active session in order to improve user guidance . . . 4
        3.1.3. UC3. As an operations or analytics service, I want timestamped connector transitions in order to diagnose delays and user-facing issues . . . 4
    3.2. Security and privacy concerns . . . 4
        3.2.1. Security concerns . . . 4
        3.2.2. Privacy concerns . . . 4
    3.3. Use of the extension . . . 4
        3.3.1. Use of the extension in OCPI 2.3.0 . . . 4
            3.3.1.1. New enum values . . . 4
            3.3.1.2. New object fields . . . 4
            3.3.1.3. New HTTP endpoints . . . 5
            3.3.1.4. New types . . . 6
            3.3.1.5. Extension discovery in OCPI 2.3.0 . . . 7
        3.3.2. Use of the extension in OCPI 3.0 . . . 7
            3.3.2.1. New enum values . . . 7
            3.3.2.2. New object fields . . . 7
            3.3.2.3. New Party Issued Objects . . . 7
            3.3.2.4. New Remote Procedure Calls . . . 7
            3.3.2.5. New types . . . 7
            3.3.2.6. Extension discovery in OCPI 3.0 . . . 8
    3.4. Compatibility and mapping strategy . . . 8
        3.4.1. Source metadata and omission . . . 8
        3.4.2. OCPP 1.5 and 1.6 mapping . . . 9
        3.4.3. OCPP 2.0.1 and OCPP 2.1 mapping . . . 9
        3.4.4. Non-OCPP sources . . . 10
        3.4.5. Additional OCPP examples . . . 10
    3.5. Forward compatibility . . . 10
4. References . . . 10

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	Contributors	Description
0.1	2026-11-02	Jens Andersson Reinier Lamers	Proposed review draft

1. Requirement terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 (RFC 2119 and RFC 8174) when, and only when, they appear in all capitals, as shown here.

2. Motivation

This extension proposes a generic way to share, over OCPI, connector-level, near-real-time status changes derived from OCPP 1.5, OCPP 1.6, OCPP 2.0.1, OCPP 2.1 and non-OCPP sources.

The OCPI Locations module is and remains the primary way for a receiver to learn about a CPO's charging facilities. It also remains the only functional module needed for the regular use case of showing chargers on a map for user discovery and navigation. However, it is not always sufficient for use cases that require precise timing and connector-level state transitions.

That gap can reduce usability for:

- mobile applications guiding users at busy sites,
- remote support and operations tooling,
- orchestration services that need to correlate user actions and connector state,
- hubs and roaming intermediaries that want a consistent status feed without requiring direct OCPP access.

This optional extension is intended only for parties that want to send or receive more detailed connector statuses, using the existing OCPI transport and security model.

This extension standardizes mappings for OCPP 1.5, OCPP 1.6, OCPP 2.0.1 and OCPP 2.1, and provides a framework in which status information from non-OCPP sources or incompatible future OCPP versions can be represented without requiring modifications to this document. Only three fields, `status`, `status_set` and `last_updated`, are required in the update payload; source metadata is optional, and omission means that origin information is not provided.

3. Domain

3.1. Use cases

3.1.1. UC1. As a CPO Platform, I want to push connector-level status transitions to partner Platforms in order to improve real-time interoperability

The CPO often already receives connector events from OCPP 1.5, OCPP 1.6, OCPP 2.0.1 or OCPP 2.1. With this extension, the CPO can forward connector status changes through OCPI to partners that do not have an OCPP connection to the Charging Station.

3.1.2. UC2. As an eMSP or app provider, I want to identify connectors that were just plugged in but have no active session in order to improve user guidance

When users open an app after plugging in, the app can highlight the likely connector candidates at that site instead of presenting a large list of all connectors in the wider area. It could even automate the entire flow. For example, the app can show a short list of connectors that were recently connected to a vehicle, but do not yet have an active session.

3.1.3. UC3. As an operations or analytics service, I want timestamped connector transitions in order to diagnose delays and user-facing issues

A consistent feed of events with `status` and `last_updated` fields helps identify where delays occur between cable connection, authorization and charging start. Note that there is no strict guarantee of every state transition being delivered, as neither OCPP nor OCPI requires updates to be queued for later delivery in case of failure.

3.2. Security and privacy concerns

3.2.1. Security concerns

This extension follows the same security model as the rest of OCPI: authentication with a secret token, and voluntary sharing of data by the sender of the data objects.

3.2.2. Privacy concerns

The data that is exchanged with this extension can provide a detailed insight into usage patterns of EVSEs. For EVSEs that are typically used by one person or a small group of people, connector status data from those EVSEs can be used to track the habits and whereabouts of those people. Connector status data senders are advised to be appropriately cautious about which EVSEs they share connector status data for, and which parties they share it with. There is no obligation on CPOs to share the connector status of every EVSE that they manage, and there is no obligation to share it with all parties that they have an OCPI connection with. CPOs are advised to only share the data where a clear use case for doing so exists, and never enable it across all EVSEs or all OCPI partners.

3.3. Use of the extension

3.3.1. Use of the extension in OCPI 2.3.0

3.3.1.1. New enum values

To the ModuleID enumeration defined in OCPI 2.3.0, the following value is added:

Module	ModuleID	Remark
Connector Statuses	connector-statuses	

3.3.1.2. New object fields

No base object fields are added by this extension.

3.3.1.3. New HTTP endpoints

This extension adds one endpoint:

- URL:
  - `{receiver_endpoint_url}/{country_code}/{party_id}/{location_id}/{evse_uid}/{connector_id}`
- Method:
  - `PUT`
- Request payload: a ConnectorStatus object
- Response payload:
  - regular OCPI response object
  - success: OCPI status code `1000`
- Query parameters:
  - none

`receiver_endpoint_url` is the module URL advertised in version details. Path identifiers use `CiString(2)` for `country_code`, `CiString(3)` for `party_id` and `CiString(36)` for `location_id`, `evse_uid` and `connector_id`.

Each `PUT` MUST include `status`, `status_set` and `last_updated`. Optional source metadata describes this update only: receivers SHOULD NOT assume the continued use of source protocol version or message type from an earlier update. The omission rules below apply independently to each update. The path identifies the connector. No identity fields are part of the update payload.

Minimal example (without the optional source metadata fields):

```
PUT /ocpi/connector-statuses/NL/ABC/loc-123/evse-456/conn-1
Content-Type: application/json
Authorization: Token xxx

{
  "status": "Preparing",
  "status_set": "OCPP_1_6",
  "last_updated": "2026-09-09T10:45:12Z"
}
```

The `status_set` field in these messages identifies the set of values, and their meanings, used in the `status` field. The three currently supported status sets are the ones used by OCPP 1.5, OCPP 1.6 and OCPP 2.0.1/2.1. Accordingly, the first example with `status Preparing` and `status_set OCPP_1_6` is read as: the Charging Station reported a status using OCPP 1.6's system of identifying statuses, and the status that it sent is `Preparing`.

Here is a more elaborate OCPP 1.6 example with source metadata:

```
PUT /ocpi/connector-statuses/NL/ABC/loc-123/evse-456/conn-1
Content-Type: application/json
Authorization: Token xxx

{
  "status": "Preparing",
  "status_set": "OCPP_1_6",
  "last_updated": "2026-08-27T10:45:12Z",
  "source_protocol_version": "OCPP_1_6",
  "source_message_type": "STATUS_NOTIFICATION"
```

```
}
```

This is read as: the Charging Station reported a status using OCPP 1.6's system of identifying statuses, the status that it sent is `Preparing`, and that status was sent using an OCPP 1.6 StatusNotification.req message.

3.3.1.4. New types

ConnectorStatus *object*

An update of a connector's status.

Property	Type	Card.	Description
status	string(64)	1	The connector status as reported by the Charging Station
status_set	ConnectorStatusSet	1	The system of classifying connector statuses that is used
last_updated	DateTime	1	A timestamp of when the change to this status occurred
source_protocol_version	SourceProtocolVersion	?	The source protocol version used by the Charging Station to report the status
source_message_type	SourceMessageType	?	The message type used by the Charging Station to report the status

ConnectorStatusSet *OpenEnum*

Identifies the system of classifying connector statuses that was used by the Charging Station to report its connector status.

Value	Description
OCPP_1_5	OCPP 1.5
OCPP_1_6	OCPP 1.6
OCPP_2	The system used by OCPP 2.0.1, OCPP 2.1 and possibly subsequent versions of OCPP that keep using the same system of connector status values and charging state values

SourceProtocolVersion *OpenEnum*

Identifies the protocol and protocol version that were used by the Charging Station to report its connector status.

Value	Description
OCPP_1_5	OCPP 1.5
OCPP_1_6	OCPP 1.6
OCPP_2_0_1	OCPP 2.0.1
OCPP_2_1	OCPP 2.1

SourceMessageType *OpenEnum*

Identifies the type of the message with which the Charging Station reported a connector status.

Value	Description
STATUS_NOTIFICATION	An OCPP StatusNotification request message
TRANSACTION_EVENT	An OCPP TransactionEventRequest message

3.3.1.5. Extension discovery in OCPI 2.3.0

A receiver MUST advertise an Endpoint with `identifier = connector-statuses` and `role = RECEIVER` in its version-details response. The Endpoint's `url` is the complete module base URL; the sender MUST append only `/{country_code}/{party_id}/{location_id}/{evse_uid}/{connector_id}` to it. The sender MUST NOT derive this URL from the Locations endpoint or append a second `connector-statuses` segment.

Example Endpoint entry:

```
{
  "identifier": "connector-statuses",
  "role": "RECEIVER",
  "url": "https://example.org/ocpi/connector-statuses"
}
```

The receiver SHOULD advertise `detailed-connector-status` in the version-details `extensions` map. Senders MUST use the advertised receiver endpoint. No Sender Interface or status-pull operation is defined.

3.3.2. Use of the extension in OCPI 3.0

3.3.2.1. New enum values

No new values for base-specification enumerations are used.

3.3.2.2. New object fields

No base object fields are added by this extension.

3.3.2.3. New Party Issued Objects

This extension introduces one Party Issued Object type:

- ModuleID: `LOCATIONS`
- Path segment for requests: `connector-status`
- Object type: `ConnectorStatus`
- Party Issued Object ID (`id` field of the `PartyIssuedObjectUpdateRequest` object): The `uid` of the EVSE containing the connector for which the status is updated.

3.3.2.4. New Remote Procedure Calls

No new RPC operations are defined by this extension.

3.3.2.5. New types

ConnectorStatus *class*

An update of a connector's status.

Property	Type	Required	Description
connector_id	CiAsciiString[1..36]	yes	The identifier of the connector whose status is updated. This field refers to an OCPI 3.0 Connector object by its `id` field. This identifier is only unique within the EVSE that the Connector is contained in; the EVSE identifier is given as the Party Issued Object ID in the `PartyIssuedObjectUpdateRequest` containing this `ConnectorStatus`.
status	AsciiString[1..64]	yes	The connector status as reported by the Charging Station
status_set	ConnectorStatusSet	yes	The system of classifying connector statuses that is used
last_updated	DateTime	yes	A timestamp of when the change to this status occurred
source_protocol_version	SourceProtocolVersion	no	The source protocol version used by the Charging Station to report the status
source_message_type	SourceMessageType	no	The message type used by the Charging Station to report the status

ConnectorStatusSet, SourceProtocolVersion, SourceMessageType

The extension adds `ConnectorStatusSet`, `SourceProtocolVersion` and `SourceMessageType`, defined as they are for OCPI 2.3.0 above.

3.3.2.6. Extension discovery in OCPI 3.0

Support SHOULD be advertised under `detailed-connector-status` in the `extensions` map. If supported only for certain parties, it SHOULD be advertised in party-specific `extensions` fields.

3.4. Compatibility and mapping strategy

3.4.1. Source metadata and omission

Both source metadata fields MAY be omitted, regardless of the actual source. If no optional fields are supplied, origin information is not provided. An omitted `source_protocol_version` MUST NOT be interpreted as OCPP 1.6 or any other source protocol.

When supplied, `source_protocol_version` MUST accurately identify the protocol, and protocol version, that the Charging Station used to report the status.

`source_message_type`, when present, identifies the type of the message that triggered the update. It is an OpenEnum, so for non-OCPP protocols or future OCPP versions, other values can be used besides `STATUS_NOTIFICATION` and `TRANSACTION_EVENT`.

`last_updated` MUST identify when the connector's status last changed to the reported status. The sender SHOULD use the original source observation timestamp, ideally provided by the Charging Station. If unavailable, the sender MAY use the backend timestamp when the status was received or observed. Forwarding or retrying an update MUST NOT

refresh `last_updated`. Receivers MUST NOT replace newer observations with older updates.

3.4.2. OCPP 1.5 and 1.6 mapping

The sender SHOULD send an OCPI connector status when:

- the sender has a Charging Station for which it wants to send detailed connector statuses, and a connected OCPI party to which it wants to send these statuses, and
- it receives a StatusNotification.req from the Charging Station, and
- the request contains a `connectorId` with a value greater than 0, and
- the request contains a status that is different from the status previously sent for the same connector via OCPI, and
- the timestamp of the StatusNotification.req is absent or not older than that in the most recent OCPP request that triggered an OCPI connector status update for the same connector.

The sender MAY apply rate limiting so as not to overload its own and the receiver's systems.

The sender MUST set the fields of the connector status as follows:

- `status` to the value of the `status` field in the StatusNotification.req message, and
- `status_set` to `OCPP_1_5` if the Charging Station uses OCPP 1.5 or `OCPP_1_6` if the Charging Station uses OCPP 1.6, and
- `source_protocol_version` either omitted, or set to the same value as `status_set`, and
- `source_message_type` either omitted, or set to `STATUS_NOTIFICATION`.

3.4.3. OCPP 2.0.1 and OCPP 2.1 mapping

The following rules apply equally to both versions. `StatusNotification.connectorStatus` reports availability; `TransactionEvent.transactionInfo.chargingState` provides charging detail. Both types of messages can trigger sending an OCPI connector status.

Accordingly, the sender SHOULD send an OCPI connector status when:

- the sender has a Charging Station for which it wants to send detailed connector statuses, and a connected OCPI party to which it wants to send these statuses, and
- it receives either a StatusNotificationRequest or a TransactionEventRequest from the Charging Station, and
- the request is a TransactionEventRequest about a transaction for which the sender knows the `evseId` and `connectorId`, or the request is a StatusNotificationRequest with a `connectorId` field with a value greater than 0, and
- the request contains a status that is different from the status previously sent for the same connector via OCPI, with the status contained in a request being the value in the `connectorStatus` field of a StatusNotificationRequest or the `transactionInfo.chargingState` field of a TransactionEventRequest, and
- the timestamp of the StatusNotificationRequest or TransactionEventRequest is not older than that in the most recent OCPP request that triggered an OCPI connector status update for the same connector.

As a result, for connectors in OCPP 2.0.1 and OCPP 2.1 Charging Stations, the values in the `status` field of OCPI connector status updates come from the union of the `ChargingStateEnumType` and `ConnectorStatusEnumType` enumerations of OCPP 2.0.1 and 2.1.

The sender MAY apply rate limiting so as not to overload its own and the receiver's systems.

The sender MUST set the fields of the connector status as follows:

- `status` to the value of the `connectorStatus` field in the OCPP request if the request is a StatusNotificationRequest, and to the value of the `transactionInfo.chargingState` field in the OCPP request if the request is a TransactionEventRequest, and
- `status_set` to `OCPP_2`, and
- `source_protocol_version` either omitted, or set to `OCPP_2_0_1` if the Charging Station uses OCPP 2.0.1 or `OCPP_2_1` if the Charging Station uses OCPP 2.1, and
- `source_message_type` either omitted, or set to `STATUS_NOTIFICATION` or `TRANSACTION_EVENT` according to which OCPP request triggered the OCPI connector status update.

3.4.4. Non-OCPP sources

A sender using a non-OCPP source MUST set `status_set` to a value that identifies the system of classifying connector status that is used.

3.4.5. Additional OCPP examples

OCPP 1.5 has its own `status_set` value, because it uses a different set of connector statuses from the one used by OCPP 1.6:

```
{
  "status": "Occupied",
  "status_set": "OCPP_1_5",
  "last_updated": "2026-09-09T10:45:12Z"
}
```

The following is a maximal example (all defined payload fields). An OCPP 2.1 TransactionEvent reports `chargingState = SuspendedEVSE` for the associated connector:

```
{
  "status": "SuspendedEVSE",
  "status_set": "OCPP_2",
  "last_updated": "2026-09-09T10:48:30Z",
  "source_protocol_version": "OCPP_2_1",
  "source_message_type": "TRANSACTION_EVENT"
}
```

3.5. Forward compatibility

Platforms that do not support this extension can safely ignore the `connector-statuses ModuleID` value. They will then have no other exposure to the types and endpoints described in this extension.

4. References

Abbreviation	Title	URL

OCPI 2.3.0	Open Charge Point Interface 2.3.0	https://evroaming.org/wp-content/uploads/2025/02/OCPI-2.3.0.pdf
OCPP 1.5	Open Charge Point Protocol: Interface description between Charge Point and Central System, document version 1.5	-
OCPP 1.6	Open Charge Point Protocol 1.6	https://openchargealliance.org/protocols/ocpp-protocols/ocpp-1-6/
OCPP 2.0.1	Open Charge Point Protocol 2.0.1 specification and schemas	https://openchargealliance.org/protocols/ocpp-protocols/
OCPP 2.1	Open Charge Point Protocol 2.1 specification and schemas	https://openchargealliance.org/protocols/ocpp-protocols/
RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	https://www.rfc-editor.org/rfc/rfc2119
RFC 8174	Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words	https://www.rfc-editor.org/rfc/rfc8174