Power Electronics Protocol over WebSocket PEP-WS

Networking and Application Protocol Requirements

Version 1.9 of 2023-02-15

Status	Released
Publisher	Vector Informatik GmbH © 2023 All rights reserved. Any distribution or copying is subject to prior written approval by Vector. Note: Hardcopy documents are not subject to change management.

Change History

Date	Name	Changes
2019-10-08	vishnt	First Draft: alpha state
2019-11-13	vishnt	Major changes in chapters Timing and Error Handling. Added chapter IMD Self Test Status
2020-01-31	vistrf	Major change of chapters Message Frames, Timings, Error Handling. Removed chapters IMD and Charging Illustration. Added JSON schemas. Adapted Charging Sequence diagram and PECC state machine proposal
2020-03-30	vistio	Keys and values are now case-sensitive and written in camelCase. Syntactical errors have been removed from json schemas. Request targetValues now includes the battery's state of charge as field batteryStateOfCharge
2020-04-17	visjnk	First chapter removed
2020-08-26	mkiefer	Added getInput and setOutput request messages Added evConnectionState message
2020-11-27	mkiefer	Clarified use of PECC operationalStatus==INOPERATIVE
2021-02-15	mkiefer	Added voltage to cableCheck request Added vehicleId to evConnectionState Clarified use of websocket PING mechanism
2021-03-05	visert	Clarified difference between driven and measured values
2021-04-10	mkiefer	Added floatValues to configuration response
2021-05-25	mkiefer	Clarified request-configuration update intervals
2021-08-15	mkiefer	Added stopCharging message
2022-01-25	mkiefer	Added chargingSession message
2022-02-15	mkiefer	Added PP supervision explanation
2022-04-15	mkiefer	Removed isolationStatus noImd
2022-05-25	mkiefer	Extended chargingSession message
2022-10-01	mkiefer	BPT Extensions (response-configuration, info-chargingSession)
2023-02-15	mkiefer	Add info-dynamicLimits message

Contents

1 Introduction

1.1 Scope

This document specifies the communication protocol between the Power Electronics Communication Controller (PECC) and the Supply Equipment Communication Controller (SECC). This Power Electronics Protocol (PEP) is designed to control and monitor the energy transfer of a power electronics used in the context of Electric Vehicle (EV) charging. Its design is based on the requirements of the ISO 15118-2 and DIN EN 61851-23.

1.2 Conventions

The following keywords within the document should be interpreted as written below:

SHALL expresses an *obligatory* / *mandatory* requirement.

SHALL NOT expresses an *absolute prohibition*.

SHOULD expresses a *recommendation* or an *advice*. Not following the recommendation should only be done when the consequences are fully understood.

SHOULD NOT expresses a discouragement. Carefully weigh the implications of deviating from the recommended procedure.

1.3 Definitions & Abbreviations

Abbreviation	Description
EV	Electric Vehicle
PEP	Power Electronics Protocol
SECC	Supply Equipment Communication Controller
PECC	Power Electronics Communication Controller
PE	Power Electronics (circuitry)
IMD	Isolation Monitoring Device

1.4 References

Reference	Description
RFC6455	"The WebSocket Protocol". http://tools.ietf.org/html/rfc6455
RFC3986	"Uniform Resource Identifier (URI): Generic Syntax". http://tools.ietf.org/html/rfc3986
RFC2616	"Hypertext Transfer Protocol-HTTP/1.1". http://tools.ietf.org/html/rfc2616
RFC3629	"UTF-8, a transformation format of ISO 10646". http://tools.ietf.org/html/rfc3629
RFC8259	"The JavaScript Object Notation (JSON) Data Interchange Format". https://tools.ietf.org/html/rfc8259

1.5 PEP Downwards Compatibility

PEP 1.1 is NOT downwards compatible to PEP 1.0.
All following versions are downwards compatible to PEP 1.1.

2 Communication Model

PEP communication is based on WebSocket as described in RFC6455. Two communication entities are defined:

- The Supply Equipment Communication Controller (SECC) acts as WebSocket client. This component governs the charging process, communicates with the EV and PECC, and controls and monitors the energy transfer using PEP. The SECC establishes a WebSocket connection and keeps it open at all times.
- The Power Electronics Communication Controller (PECC) acts as WebSocket server. This component manages and provides an interface to the power electronics (i.e., the physical device) itself. It is primarily responsible for executing and answering requests sent by the SECC. In addition the PECC can also send requests to the SECC, e.g., to control SECC outputs. The PECC waits for the SECC to establish a WebSocket connection.

PEP defines two communication patterns:

- A request-reply pattern implemented by request, response and error messages. This mechanism is intended for reliable control of a remote device such as the power electronics' contactors.
- A one-way pattern implemented by info messages. This mechanism is intended for letting the other entity know about an internal state.

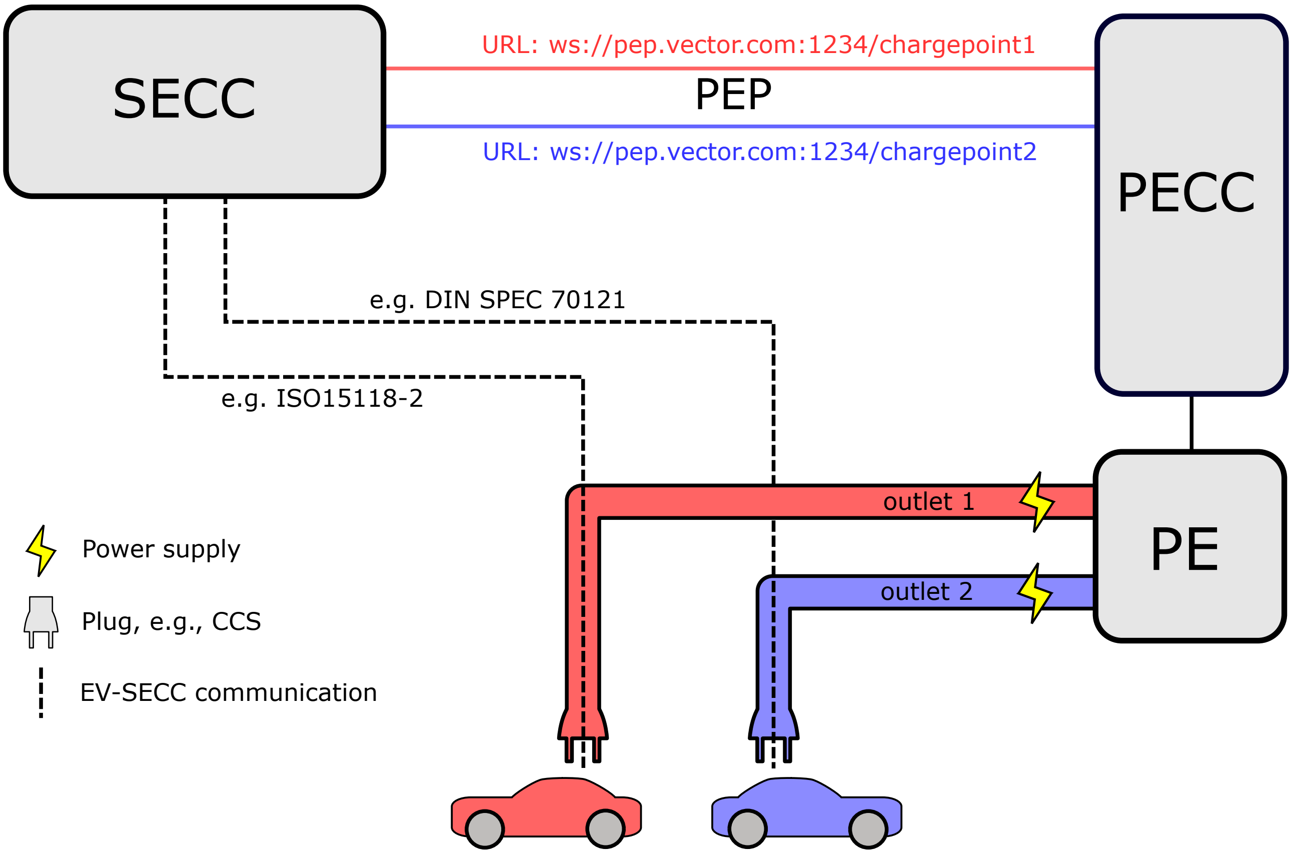
2.1 The Connection URL

To initiate a WebSocket connection, the SECC needs a URL (RFC3986) to connect to. This PEP endpoint URL is called the *connection URL*.

A charging session (in the sense of ISO 15118) corresponds to exactly one *charge point*. A charge point denotes a power outlet that belongs to at most one charging session. Conceptually, a single Power Electronics (PE) managed by its PECC could provide energy for multiple charge points simultaneously.

PEP defines no method of addressing different charge points. This means that a single connection URL corresponds to exactly one charge point. If a single PECC wants to manage multiple charge points, it SHALL offer multiple connection URLs.

An example connection URL is “ws://pep-server.vector.com:1234/chargepoint1”.



2.2 Reconnection Attempts

When the WebSocket connection is lost, the SECC SHOULD try to reconnect once every PEP_WS_RECONNECT_INTERVAL (10000 ms).

If the reconnection attempt succeeds, the PECC SHALL be ready to answer requests and send the status messages periodically. The SECC SHALL be able to process requests received as soon as it connects to the PECC.

2.3 PEP-specific WebSocket Data

The WebSocket header "Sec-WebSocket-Protocol" header SHALL be set to "pep<PEP version>".

PEP described in this document uses the following header: "Sec-WebSocket-Protocol: pep1.5".

2.4 Synchronicity

There SHALL be at most one request pending for each communication direction. This means that a communication controller (either SECC or PECC) SHALL NOT send request messages unless previous request messages sent by this communication controller have been responded to or have timed out.

A request has been responded to if a reply (response or error message) with the same sequenceNumber as sent in the request is received within PEP_REQUEST_TIMEOUT (500 ms).

A request has timed out if it has not been responded to within PEP_REQUEST_TIMEOUT (500 ms).

Informational messages can be sent at all times, in particular in between processing a request message and sending the response message.

3 Message Frames

3.1 Message Frame Structure

PEP messages consist of JavaScript Object Notation (JSON) encoded UTF-8 strings. Each information consists of a key-value-pair. String values of both keys and values are case-sensitive and SHALL follow the camelCase notation used in the associated json schemas.

The following two fields exist in every message:

type: This field describes the overall purpose of a message. Four types exist: request, response, error and info. The type is a classification of a message at communication model level. The messageType defines the layout of the overall message. In contrast to all other types, messages with type "info" do not contain a sequence number.

kind: This field refines the respective type and describes the action to be executed or component that is affected by the message. The kind is a classification and description of a message at application level. The kindType defines the layout of the payload field.

A message is a syntactically valid PEP message if and only if the respective JSON schema validates successfully against it. JSON schemas for each message are listed in the appendix Section PEP JSON Schemas.

If a valid request is received, it SHALL be answered with its respective response message. If the request received is

- syntactically (e.g., schema validation failed) or
- semantically (e.g., requested voltage exeeds limits) invalid, an error message SHALL be sent.

In the following subsections, examples given in the message descriptions show the complete message, i.e., with the type, kind and sequenceNumber (where applicable), not just the payload object itself.

3.2 Request Messages

Request messages are sent by both the SECC and the PECC. They have the following fields:

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the message and layout of the payload field.
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Payload corresponding to the message kind. Empty object if not required.

3.2.1 configuration

This message is used to request the current configuration of the power electronics. It could be sent anytime and multiple times. This implies that it is up to the SECC implementation if these values are requested regularly and when they are sent to the EV (e.g.: on startup of the SECC/ before every charging cycle/regularly during a charging cycle/. . .)

Answered with response - configuration.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "configuration".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "request",
  "kind": "configuration",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.2.2 cableCheck

This message is used to request a high voltage isolation check, usually at the beginning of a charging process. It contains the suggested isolation check voltage, in case of CCS this is the maximum voltage of the EV. For CHAdeMO this is the voltage defined in the specification. The result is reported with info - status.

Answered with response - cableCheck.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "cableCheck".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Payload containing the suggested isolation check voltage.

Payload fields:

Payload Field Name	Field Type	Physical Unit	Description
voltage	number	V/Volts	Suggested isolation check voltage.

Example message:

```
{
  "type": "request",
  "kind": "cableCheck",
  "sequenceNumber": 458238,
  "payload": {
    "voltage": 500
  }
}
```

3.2.3 targetValues

This message is used to instruct the power electronics to drive its outputs with the requested values for voltage and current. Furthermore, the current charging process state of the EV and the battery's state of charge are reported with the chargingState and batteryStateOfCharge field, respectively.

Answered with response - targetValues.

If the requested voltage exceeds the power electronics' internal voltage limit, this message SHALL be answered with an error message with the errorCategory set to valueError. In this case, the communication SHALL NOT stop, i.e., the Websocket connection is kept open and other actions (e.g., contactor state changes) SHALL NOT be executed as a direct consequence of this message.

If the requested voltage is within the power electronics' internal voltage limit but the requested current could not be supplied, this message SHALL be answered with response - targetValues. In this case, the output should be driven with the requested voltage and the maximum current the power electronics is able to supply with respect to both power and electrical current limits (degraded performance).

If this message is received at an inappropriate instant (e.g., if contactors are open), it SHALL be ignored.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "targetValues".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Payload containing the requested values for voltage and current.

Payload fields:

Payload Field Name	Field Type	Physical Unit	Description
targetVoltage	number	V/Volts	Voltage to be driven by the power electronics.
targetCurrent	number	A/Amperes	Current to be driven by the power electronics
batteryStateOfCharge	number	%	The vehicles' battery state of charge. Values are in the range from 0 to 100, inclusive.
chargingState	chargingStateType		Current charging state of the EV. See chargingStateType for details.

Example message:

```
{
  "type": "request",
  "kind": "targetValues",
  "sequenceNumber": 458238,
  "payload": {
    "targetVoltage": 600,
    "targetCurrent": 21,
    "batteryStateOfCharge": 50,
    "chargingState": "charge"
  }
}
```

3.2.4 contactorsStatus

This message is used to instruct the power electronics to open or close its contactors.

Answered with response - contactorsStatus.

If the actual state equals the one requested, this message SHALL be ignored.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "contactorsStatus".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Payload containing the requested new state of the contactors.

Payload fields:

Payload Field Name	Field Type	Description
contactorsStatus	contactorsStatusType	New state of the contactors.

Example message:

```
{
  "type": "request",
  "kind": "contactorsStatus",
  "sequenceNumber": 458238,
  "payload": {
    "contactorsStatus": "closed"
  }
}
```

3.2.5 reset

This message is used to reset the power electronics' status to a valid, well-defined standby state. In this state, the contactors are open, the driven voltage and current are set to 0, the PECC sends the periodic info - status messages and is able to receive new requests, i.e., the operationalStatus is "operative". See Error Handling for further details.

This message does not affect the WebSocket communication, i.e., the WebSocket connection is not closed and the PECC is not rebooted.

This message may be received anytime, in particular prior to, during or after a charging process.

Answered with response - reset.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "reset".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "request",
  "kind": "reset",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.2.6 getInput

This message is used by the PECC to read inputs from the SECC. Multiple inputs can be requested with a single message. Available input identifiers depend on the hardware, refer to the SECC hardware manual.

Answered with response - getInput.

If one of the requested input identifiers is invalid, this message SHALL be answered with an error message with the errorCategory set to valueError.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "getInput".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Contains the requested inputs.

Payload fields:

Payload Field Name	Field Type	Description
inputIdentifiers	array	Array of input identifiers to be requested.

Example message:

```
{
  "type": "request",
  "kind": "getInput",
  "sequenceNumber": 458238,
  "payload": {
    "inputIdentifiers" : ["d1","a1","t3"]
  }
}
```

3.2.7 setOutput

This message is used by the PECC to set outputs at the SECC. Multiple outputs can be set with a single message. Available output identifiers depend on the hardware, refer to the SECC hardware manual.

Answered with response – setOutput.

If one of the requested output identifiers or values is invalid, this message SHALL be answered with an error message with the errorCategory set to valueError.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "setOutput".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Contains the outputs to be set.

Payload fields:

Payload Field Name	Field Type	Description
outputValues	JSON	Key/value pairs of outputs and values to be set. Keys are of type string and refer to the identifier of the output. Type of a value depends on the corresponding output. Refer to the SECC hardware manual.

Example message:

```
{
  "type": "request",
  "kind": "setOutput",
  "sequenceNumber": 458238,
  "payload": {
    "outputValues" : {
    "d1": 1,
    "d2": 0
    }
  }
}
```

3.2.8 stopCharging

This message is used by the PECC to request a graceful termination of the running charging session.

Answered with response – stopCharging.

If no charging session is running, or it cannot be terminated in the current state, this message SHALL be ignored.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "request".
kind	kindType	Identifies the kind of the request. Set to "stopCharging".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "request",
  "kind": "stopCharging",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3 Response Messages

Response messages are sent by both the SECC and the PECC. They have the following fields:

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Set to the kind of the request this message responds to.
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Payload of the message kind. Empty object {} if not required.

3.3.1 configuration

This message is used to answer request - configuration messages. It provides the current configuration of the power electronics.

The fields *limitPowerMin*, *limitDischargeCurrentMin*, *limitDischargeCurrentMax*, *limitDischargePowerMin* and *limitDischargePowerMax* are only required for ISO 15118–20 BPT.

Note: Discharge values are always negative.

It is recommended to set *floatValues* to *true*. Setting it to *false* or omitting the *floatValues* key leads to a replacement of all values of type *number* (floating point) with values of type *integer* in every JSON message schema. This entails a loss of precision and is only supported for compatibility reasons.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to “response”.
kind	kindType	Identifies the kind of the request. Set to “configuration”.
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Contains the requested configuration values.

Payload fields:

Payload Field Name	Field Type	Physical Unit	Description
firmwareVersion	string		Firmware version of the power electronics software
manufacturer	string		Power electronics manufacturer
limitVoltageMin	number	V/Volts	Minimum voltage supported by the power electronics
limitVoltageMax	number	V/Volts	Maximum voltage supported by the power electronics
limitCurrentMin	number	A/Amperes	Minimum current supported by the power electronics
limitCurrentMax	number	A/Amperes	Maximum current supported by the power electronics
limitPowerMin	number	W/Watts	Minimum power output supported by the power electronics
limitPowerMax	number	W/Watts	Maximum power output supported by the power electronics
limitDischargeCurrentMin	number	W/Watts	Minimum discharge current supported by the power electronics (negative value)
limitDischargeCurrentMax	number	W/Watts	Maximum discharge current supported by the power electronics (negative value)
limitDischargePowerMin	number	W/Watts	Minimum discharge power supported by the power electronics (negative value)
limitDischargePowerMax	number	W/Watts	Maximum discharge power supported by the power electronics (negative value)
floatValues	boolean		Use floating point numbers for voltages and currents

Example message:

```
{
  "type": "response",
  "kind": "configuration",
  "sequenceNumber": 458238,
  "payload": {
    "firmwareVersion": "pe_1.0.2",
    "manufacturer": "pe_manufacturer1",
    "limitVoltageMin": 0,
    "limitVoltageMax": 700,
    "limitCurrentMin": 0,
    "limitCurrentMax": 50,
    "limitPowerMin": 0,
    "limitPowerMax": 30000,
    "limitDischargeCurrentMin": 0,
    "limitDischargeCurrentMax": -30,
    "limitDischargePowerMin": 0,
    "limitDischargePowerMax": -15000,
    "floatValues": true
  }
}
```

3.3.2 cableCheck

This message is used to answer request - cableCheck messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "configuration".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Contains the requested configuration values.

Example message:

```
{
  "type": "response",
  "kind": "cableCheck",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3.3 targetValues

This message is used to answer request - targetValues messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "targetValues".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "response",
  "kind": "targetValues",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3.4 contactorsStatus

This message is used to answer request - contactorsStatus messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "contactorsStatus".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "response",
  "kind": "contactorsStatus",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3.5 reset

This message is used to answer request - reset messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "reset".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "response",
  "kind": "reset",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3.6 getInput

This message is used to answer request – getInput messages. It returns the values corresponding to the requested input identifiers.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "getInput".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Values of the requested input identifiers.

Payload fields:

Payload Field Name	Field Type	Description
inputValues	JSON	Key/value pairs of input identifiers and values

Example message:

```
{
  "type": "response",
  "kind": "getInput",
  "sequenceNumber": 458238,
  "payload": {
  "inputValues": {
      "d1": 1,
      "a1": 5.2,
      "t3": 40
    }
  }
}
```

3.3.7 setOutput

This message is used to answer request - setOutput messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "setOutput".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "response",
  "kind": "setOutput",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.3.8 stopCharging

This message is used to answer request - stopCharging messages. It acknowledges the reception and processing.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "response".
kind	kindType	Identifies the kind of the request. Set to "stopCharging".
sequenceNumber	integer	Used to match request and response/error messages.
payload	JSON	Empty object {}.

Example message:

```
{
  "type": "response",
  "kind": "stopCharging",
  "sequenceNumber": 458238,
  "payload": {}
}
```

3.4 Error Messages

3.4.1 error

Error messages are sent as answers to request messages. They SHALL be sent if a problem with the request itself or the requested action occurs. Possible problems include failed syntax validation or invalid requested values. The most appropriate errorCategory should be chosen and the errorDetails field filled with an additional error description.

If the sequence number of a request could not be determined (e.g., if the JSON could not be parsed), it SHALL be set to the special value 0. See Sequence Numbers for details.

All error messages have the following fields:

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "error".
kind	kindType	Identifies the kind of the request. Set to the respective kind from the request this message answers.
sequenceNumber	integer	Used to match request and response/error messages. Set to 0 if sequence number of the request could not be determined.
payload	JSON	Contains details of the error.

Payload fields:

Payload Field Name	Field Type	Description
errorCategory	errorCategoryType	Category of the error described further in the errorDetails field.
errorDetails	string	Detailed description or "" (empty string).

Example message:

```
{
  "type": "error",
  "kind": "cableCheck",
  "sequenceNumber": 458238,
  "payload": {
    "errorCategory": "format",
    "errorDetails": "JSON schema validation failed at line 5."
  }
}
```

3.5 Informational Messages

Informational messages are sent by both SECC and PECC and have the following fields:

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the message and layout of the payload field.
payload	JSON	Payload of the message kind. Empty object {} if not required.

3.5.1 event

This message is used to inform the SECC about an event that happened at the power electronics. It is intended to send vendor-specific messages. The detailed event description is given in the payload field "eventDetails".

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the information. Set to “event”.
payload	JSON	Contains details of the event.

Payload fields:

Payload Field Name	Field Type	Description
eventDetails	string	Event description.

Example message:

```
{
  "type": "info",
  "kind": "event",
  "payload": {
    "eventDetails": "vendor specific description of an event"
  }
}
```

3.5.2 status

This message is used to report the current status of the power electronics, including currently measured values and other status information, e.g., the contactorsStatus. The field *isolationStatus* has to be set according to ISO 15118-2.

Whenever the PECC encounters a problem and cannot operate normally, it SHALL set *operationalStatus* to *INOPERATIVE*, which SHALL cause the SECC to abort the charging process and prevent further charging sessions until the PECC returns to an *OPERATIVE* state.

In this message, the PECC is able to differentiate driven values from actual measured ones with the fields *drivenVoltage/-Current* and *measuredVoltage/-Current*. This becomes handy when no power is driven but the currently measured values are of interest for the charging phase. This is for example the case during WeldingDetection in DIN/ISO charging. The driven values currently are only for informational purpose and may be set equal to the measured values. The measured values however are the values reported to the EV and SHALL always present the actual values from PECC's perspective.

The status message SHALL be sent periodically every PEP_STATUS_UPDATE_INTERVAL (200 ms). It may be used by the SECC to detect that the PECC is unresponsive.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the information. Set to “status”.
payload	JSON	Contains the current status of the power electronics.

Payload fields:

Payload Field Name	Field Type	Physical Unit	Description
measuredVoltage	number	V/Volts	Voltage measured at the outlet.
measuredCurrent	number	A/Amperes	Current measured at the outlet.
drivenVoltage	number	V/Volts	Voltage driven at the output.
drivenCurrent	number	A/Amperes	Current driven at the output.
temperature	number	°C/Degrees Celsius	Current temperature of the power electronics.[1]
contactorsStatus	contactorsStatusType		Current status of the contactors.
isolationStatus	isolationStatusType		Current isolation status of the cable.
operationalStatus	operationalStatusType		Current operational status of the power electronics as a whole.

Example message:

```
{
  "type": "info",
  "kind": "status",
  "payload": {
    "measuredVoltage": 599,
    "measuredCurrent": 19,
    "drivenVoltage": 600,
    "drivenCurrent": 20,
    "temperature": 34.72,
    "contactorsStatus": "closed",
    "isolationStatus": "valid",
    "operationalStatus": "operative"
  }
}
```

3.5.3 evConnectionState

This message is used to report the connection state of the EV. If available, the vehicle identifier is transmitted when the connection state changes to *connected*. The *vehicleId* field is not present for all other values of *evConnectionState*.

The *evConnectionState* message SHALL be sent by the SECC whenever the EV connection state changes. Safety critical decisions SHALL NOT be made based on this purely informational message (see CP/PP supervision). Also, the charging cycle is controlled by the contactorsStatus and targetValues messages.

[1] This value is used for diagnostic/monitoring purposes only. The SECC does not act on this value.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the information. Set to “evConnectionState”.
payload	JSON	Contains the current EV connection state.

Payload fields:

Payload Field Name	Field Type	Description
evConnectionState	evConnectionStateType	Current EV connection state.
vehicleId	string	Identifier of the connected vehicle. Only present if evConnectionState==CONNECTED.

Example message:

```
{
  "type": "info",
  "kind": "evConnectionState",
  "payload": {
    "evConnectionState": "connected",
    "vehicleId": "AB:CD:12:34:56:78"
 }
}
```

3.5.4 chargingSession

This message is used to provide information about the current charging session.

The *chargingSession* message SHALL be sent by the SECC whenever the information contained in the message changes. It is not required to send all the fields in the payload, so only fields with updated information need to be transmitted.

Note: Discharge values are always negative.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the information. Set to “chargingSession”.
payload	JSON	Contains information about the current charging session.

Payload fields:

Payload Field Name	Field Type	Description
chargingProfileMaxPowerLimitWatts	number	Maximum power in Watts contained in the current charging profile.
timeToFullSocSeconds	number	Remaining time in seconds to full SOC.
evMinVoltageVolts	number	Minimum voltage the EV can handle in Volts.
evMaxVoltageVolts	number	Maximum voltage the EV can handle in Volts.
evMinCurrentAmperes	number	Minimum current the EV can handle in Amperes.
evMaxCurrentAmperes	number	Maximum current the EV can handle in Amperes.
evMinPowerWatts	number	Minimum power the EV can handle in Watts.
evMaxPowerWatts	number	Maximum power the EV can handle in Watts.
evMinDischargeCurrentAmperes	number	Minimum discharge current (negative value) the EV can handle in Amperes.
evMaxDischargeCurrentAmperes	number	Maximum discharge current (negative value) the EV can handle in Amperes.
evMinDischargePowerWatts	number	Minimum discharge power (negative value) the EV can handle in Watts.
evMaxDischargePowerWatts	number	Maximum discharge power (negative value) the EV can handle in Watts.

Example messages:

```
{
  "type": "info",
  "kind": "chargingSession",
  "payload": {
    "chargingProfileMaxPowerLimitWatts": 150000.0
  }
}
```

```
{
  "type": "info",
  "kind": "chargingSession",
  "payload": {
    "evMinVoltageVolts": 0,
    "evMaxVoltageVolts": 400,
    "evMinCurrentAmperes": 0,
    "evMaxCurrentAmperes": 350,
    "evMinPowerWatts": 0,
    "evMaxPowerWatts": 125000,
    "evMinDischargeCurrentAmperes": 0,
    "evMaxDischargeCurrentAmperes": -100,
    "evMinDischargePowerWatts": 0,
    "evMaxDischargePowerWatts": -20000,
    "chargeMode": "dynamicBpt"
  }
}
```

3.5.5 dynamicLimits

This message is used to provide dynamic limits, which represent the currently applicable (based on temporary limitations of the PE) limits, which are communicated during a running charging session (e.g. CurrentDemand/ChargeLoop) to the EV. In contrast, the limits sent in the `response-configuration` are sent as static limits to the EV (e.g. during ChargeParameterDiscovery).

The limits are kept by the SECC until the connection to the PECC is lost. If no dynamic limits are defined for some values, the static limits are applied.

It is not required to send all the fields in the payload, so only fields with updated information need to be transmitted.

Note: Discharge values are always negative. Minimum current values are missing since they cannot be reported to the EV in CurrentDemand/ChargeLoop.

Field Name	Field Type	Description
type	messageType	Identifies the type of the PEP message. Set to "info".
kind	kindType	Identifies the kind of the information. Set to “dynamicLimits”.
payload	JSON	Contains information about the currently applicable limits of the PE.

Payload fields:

Payload Field Name	Field Type	Description
limitVoltageMin	number	Minimum voltage in Volts.
limitVoltageMax	number	Maximum voltage in Volts.
limitCurrentMax	number	Maximum current in Amperes.
limitPowerMin	number	Minimum power in Watts.
limitPowerMax	number	Maximum power in Watts.
limitDischargeCurrentMax	number	Maximum discharge current (negative value) in Amperes.
limitDischargePowerMin	number	Minimum discharge power (negative value) in Watts.
limitDischargePowerMax	number	Maximum discharge power (negative value) in Watts.

Example messages:

```
{
  "type": "info",
  "kind": "dynamicLimits",
  "payload": {
    "limitPowerMax": 35000,
    "limitDischargePowerMax": -5000
  }
}
```

3.6 Sequence Numbers

Sequence numbers are used to match requests to their respective replies, either response or error messages. The sequence number sent in requests SHALL be incremented for each request by 1, its value increases monotonically. If the valid range is exceeded, counting SHALL restart from 1. The sequence number in replies SHALL be set to the same value of the sequenceNumber field in the request it answers. There are two sequence numbers, one for each request direction. They are increased independently.

If the sequence number from a request message could not be determined (e.g., if the JSON parsing fails), it SHALL be set to the special value 0.

Regular sequence numbers are valid in the range from 1 to 2147483647 (2^{31} - 1) inclusive.

Field Name	Field Type	Description
sequenceNumber	integer	Valid in the range between 1 and 2^{31} - 1.

3.7 Type Definitions

3.7.1 messageType

type	Description
info	Sent by both SECC and PECC. An info message does not get responded to.
request	Sent by both SECC and PECC. Gets responded to with a response or error.
response	Sent by both SECC and PECC. Response to a valid request.
error	Sent by both SECC and PECC. Response to an invalid request.

3.7.2 kindType

type	Description
configuration	Identifies messages required for the configuration report mechanism.
cableCheck	Identifies messages required for an isolation check.
targetValues	Identifies messages required for supplying of a voltage and current.
contactorsStatus	Identifies messages required for a contactor state change.
reset	Identifies messages required for the reset mechanism.
getInput	Identifies messages required for input readout.
setOutput	Identifies messages required for setting outputs.
error	Identifies messages required for error handling.
event	Identifies messages that report a generic, vendor-specific event.
status	Identifies messages that report the current status.
evConnectionState	Identifies messages that report the current EV connection state.
stopCharging	Identifies messages sent by the PECC to stop charging.
chargingSession	Identifies messages containing information about the current charging session.

3.7.3 contactorsStatusType

type	Description
open	Contactors are open.
closed	Contactors are closed.

3.7.4 isolationStatusType

See ISO 15118-2 for details.

type	Description
invalid	No isolation test has been carried out yet.
valid	Isolation test completed without warning or fault.
warning	Isolation test resulted with a measured isolation resistance below the warning level defined in IEC CDV 61851-23.
fault	Isolation test resulted with a measured isolation resistance below the fault level defined in IEC CDV 61851-23.

3.7.5 operationalStatusType

type	Description
operative	Power electronics is able to supply voltage or power. Normal state of operation.
inoperative	Power electronics is not able to supply voltage or power.

3.7.6 chargingStateType

type	Description
standby	No charging process started yet.
preCharge	EV is pre-charging.
charge	EV is charging.
postCharge	EV completed charging and is in a post-charging state. EV welding detection may be conducted in this state.

3.7.7 errorCategoryType

type	Description
format	Message format is incorrect, i.e., the JSON schema validation fails.
value	Invalid input/output identifier. or Requested voltage could not be supplied. Note: It is not a valueError if the requested voltage could be supplied but the requested current could not be supplied. In this case, the output should be driven with the requested voltage and the electrical current limit (degraded performance).
inoperative	The power electronics is not able to execute the request due to its current operationalStatus.
internal	An error internal to PECC or SECC occurred which prevented the processing of the request.
generic	Any other error not covered by other categories in this table.

3.7.8 evConnectionStateType

type	Description
disconnected	No connection to an EV (not plugged in).
connected	EV connected (plugged in).
energyTransferAllowed	Energy transfer to the EV is allowed.
error	An error has occured, e.g. short circuit between control pilot and protective conductor, SECC inoperative, ...

4 Timing Recommendations

The communication between an EV and SECC SHALL satisfy strict timing requirements. These requirements are detailed in the respective standard, e.g., ISO 15118.

PEP itself does not require neither SECC nor PECC to adhere to these timing requirements. However, it does recommend compliance with the following timings. If these timing recommendations are met, a charging session is not aborted due to a timing violation caused by the communication between SECC and PECC. This applies to charging sessions using ISO 15118-2 or DIN SPEC 70121 for EV-SECC communication.
SECC and PECC SHOULD comply with the following PEP timings:

Name	Value [ms]
PEP_REQUEST_TIMEOUT	500
PEP_WS_RECONNECT_INTERVAL	10000
PEP_STATUS_UPDATE_INTERVAL	200
PEP_SECC_UNRESPONSIVE_TIMEOUT	5000

In addition, the power electronics SHOULD satisfy the following non-PEP timings :

Name	Value [ms]	Description	Derived from
PE_CABLE_CHECK_TIME	37000	Time needed to perform a cable check. Measurement start: cableCheck request is received by PECC. Measurement end: Result is available at PECC. The time needed for reporting and processing a status update is taken into account.	V2G_SECC_CableCheck_Performance_Time
PE_PRE_CHARGE_TIME	4000	Time needed to pre-charge. Measurement start: targetValues request with the "chargingState" field set to "preCharge" is received by the PECC. Measurement end: The measured output voltage equals the requested voltage with a maximum deviation according to IEC CDV 61851-23. Note that multiple pre-charging requests may be received. The timer is reset for each pre-charge request. The time needed for reporting and processing a status update is taken into account.	V2G_SECC_PreCharge_Performance_Time

5 Error Handling

The goal of PEP error handling is the prevention of a failure condition or the recovery from an invalid state (possibly distributed across SECC and PECC) to a safe, well-defined and stable state.

This well-defined state is called the *standby* state and characterized by the following properties visible to the SECC:

Property	Physical Unit/Type	Description	Value
measuredVoltage	V/Volts	Voltage measured at the outlet.	Within local regulations for touch voltage. E.g.: IEC 61851-1: $\leq$ 60 V
measuredCurrent	A/Amperes	Current measured at the outlet.	0[2]
drivenVoltage	V/Volts	Voltage driven at the output.	0
drivenCurrent	A/Amperes	Current driven at the output.	0
temperature	°C/Degrees Celcius	Current temperature at the outlet.	within PE operation limits
contactorsStatus	contactorsStatusType	Current status of the contactors.	open
isolationStatus	isolationStatusType	Current isolation status of the cable.	<neglected>
operationalStatus	operationalStatusType	Current operational status of the power electronics as a whole.	operative

If an invalid state is detected by the PECC, the operationalStatus SHALL be set to "inoperative". If the error condition is remedied, the operationalStatus SHALL be set to “operative”.

The following conditions SHALL lead to a transition of the PECC and power electronics to the standby state:

- It is detected that the SECC is unresponsive. This may be due to a networking failure or SECC-internal error. The SECC is considered unresponsive if it does not respond to messages within PEP_SECC_UNRESPONSIVE_TIMEOUT (5000 ms). This may be implemented using WebSocket Ping and Pong messages RFC6455. THE SECC SHALL NOT use this mechanism to detect an unresponsive PECC, for this purpose the periodic info - status message may be used.
- PECC receives a reset request. This is used by the SECC to handle errors from the EV-SECC communication such as message timeouts.
- PECC receives a contactorsStatus request with contactorsStatus set to "open" while the current (i.e., actual) contactorsStatus is "closed".

Note that it is possible to implement a *transition phase*, i.e., a period between the invalid and standby state. This allows for example the discharging of capacities. While in this transition phase, the operationalStatus SHALL be set to "inoperative" and requests received SHALL be answered with an error message with the errorCategory set to "inoperative". The periodic status update is unaffected and sent continuously.

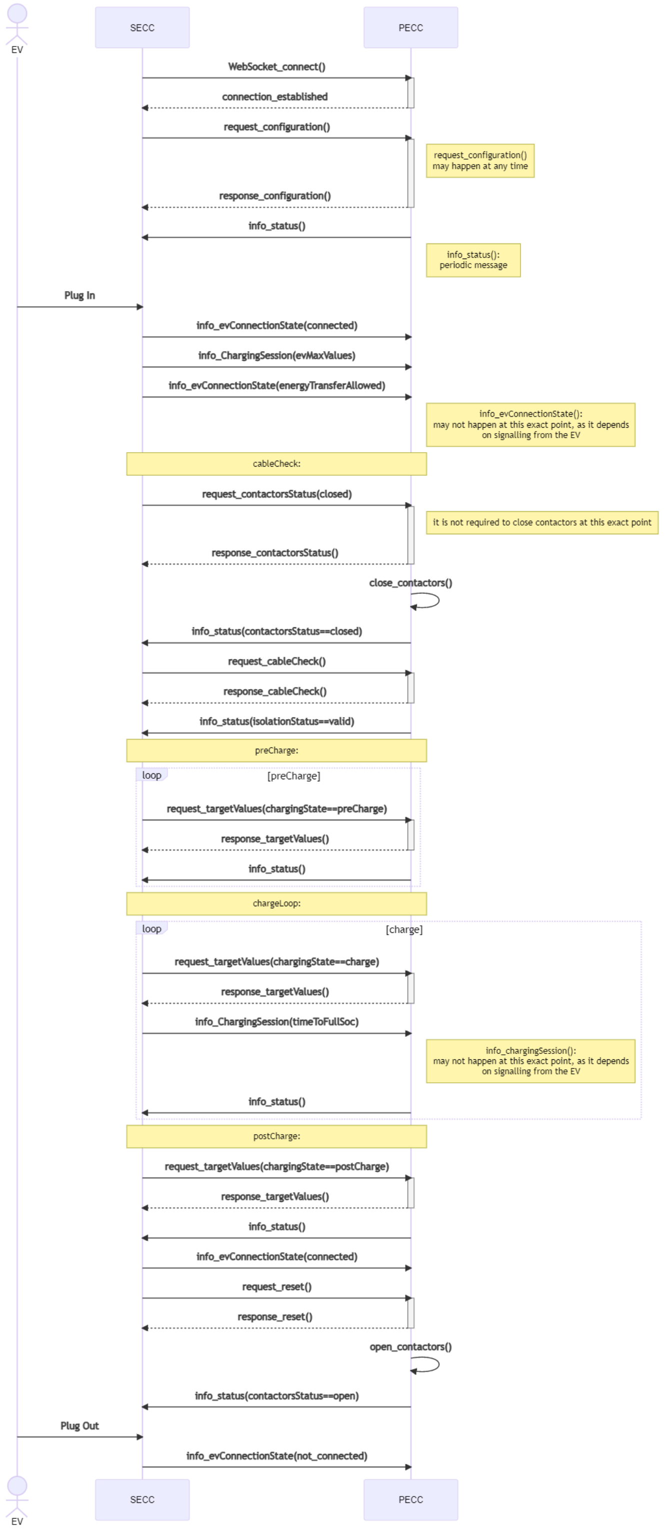
If requests could not be processed by the PECC due to the CP/PP supervision, an error message

[2] may not actually be zero, but the value is transmitted to the EV and SHALL meet applicable requirements.

SHALL be sent with the errorCategory set to "internal".

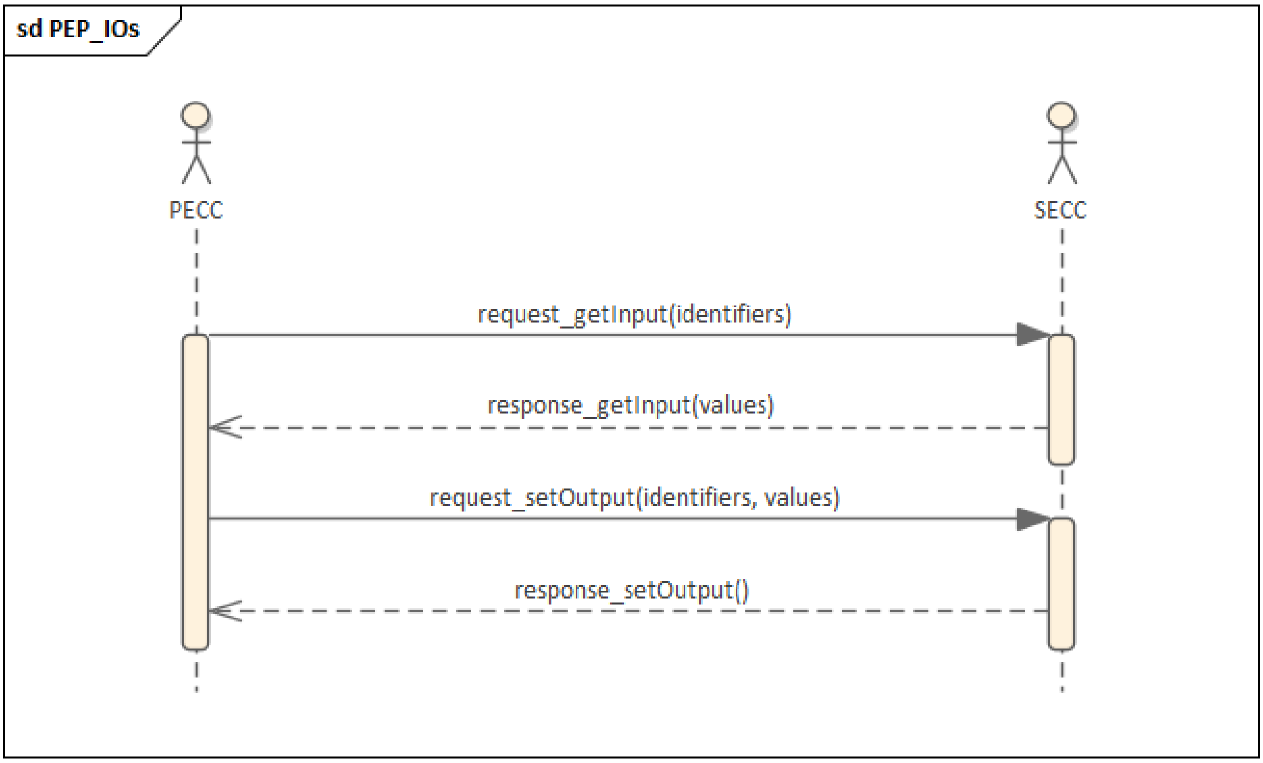
6 Example Scenarios

6.1 Charging Sequence


EV
SECC
PECC
WebSocket_connect()
connection_established
request_configuration()
request_configuration()
may happen at any time
response_configuration()
info_status()
info_status():
periodic message
Plug In
info_evConnectionState(connected)
info_ChargingSession(evMaxValues)
info_evConnectionState(energyTransferAllowed)
info_evConnectionState():
may not happen at this exact point, as it depends
on signalling from the EV
cableCheck:
request_contactorsStatus(closed)
it is not required to close contactors at this exact point
response_contactorsStatus()
close_contactors()
info_status(contactorsStatus==closed)
request_cableCheck()
response_cableCheck()
info_status(isolationStatus==valid)
preCharge:
loop
[preCharge]
request_targetValues(chargingState==preCharge)
response_targetValues()
info_status()
chargeLoop:
loop
[charge]
request_targetValues(chargingState==charge)
response_targetValues()
info_ChargingSession(timeToFullSoc)
info_chargingSession():
may not happen at this exact point, as it depends
on signalling from the EV
info_status()
postCharge:
request_targetValues(chargingState==postCharge)
response_targetValues()
info_status()
info_evConnectionState(connected)
request_reset()
response_reset()
open_contactors()
info_status(contactorsStatus==open)
Plug Out
info_evConnectionState(not_connected)
EV
SECC
PECC

- The SECC opens a WebSocket connection to the PECC.
- As soon as the WebSocket connection is established, the info – status message is sent periodically.
- The configuration can be requested once or periodically by the SECC at any time.
- The EV triggers the charging process.
- The cable check result "valid" is needed to continue.
- In the pre-charging phase, multiple targetValues requests may be received by the PECC.
- The charging phase begins as soon as a targetValues request is received with chargingState set to "charge".
- The charging process ends with a targetValues request with the chargingState set to "postCharge". The requested voltage and current is set to 0. The EV may then perform an optional welding detection of its internal contactors. After this phase, the reset request is sent and the EV unplugs.
- The evConnectionState message gets sent when the EV signals the corresponding state.

6.2 Input/Output Control


sd PEP_IOs
PECC
SECC
request_getInput(identifiers)
response_getInput(values)
request_setOutput(identifiers, values)
response_setOutput()

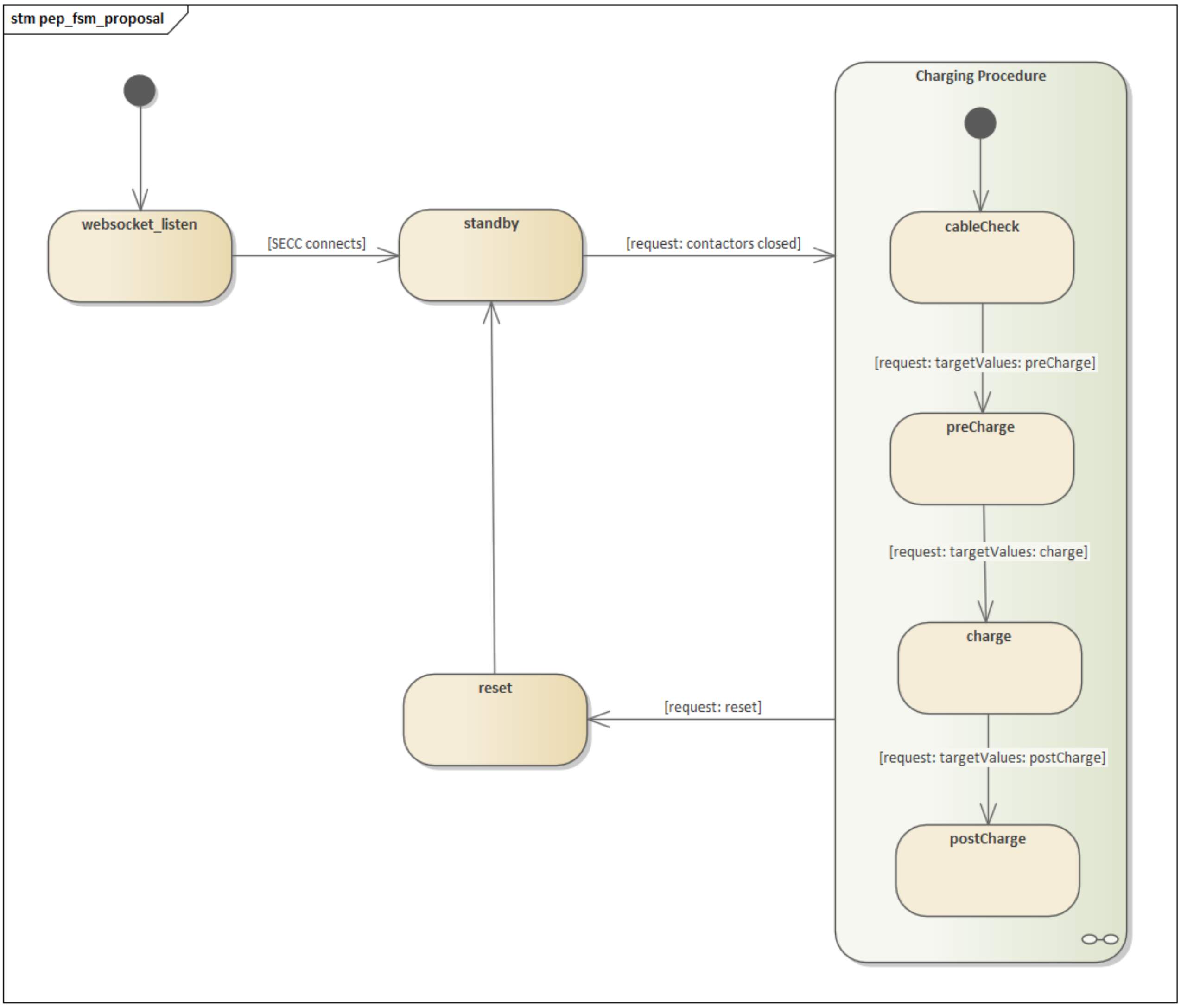
- Accessing inputs and outputs is not part of the charging process.
- getInput and setOutput requests can be sent anytime.

7 PECC State Machine Proposal

The following state machine is a suggestion and meant as implementation hint. It is NOT required to implement the PECC in exactly this manner. Note that various mechanisms and PEP messages are not included, such as the error messages and the PECC-internal error handling itself.

Note the following crucial properties:

- A charging procedure begins with the contactorsStatus request with contactorsStatus set to "closed".
- The current charging phase (preCharge, charge, postCharge) is reported by the targetValues request.
- The reset request may be received anytime while charging. The "reset" state shown is meant as a cleanup and reinitialization state. See Error Handling for details. In addition, the reset request is the normal (i.e., non-erroneous) transition from the postCharge state to the standby state.



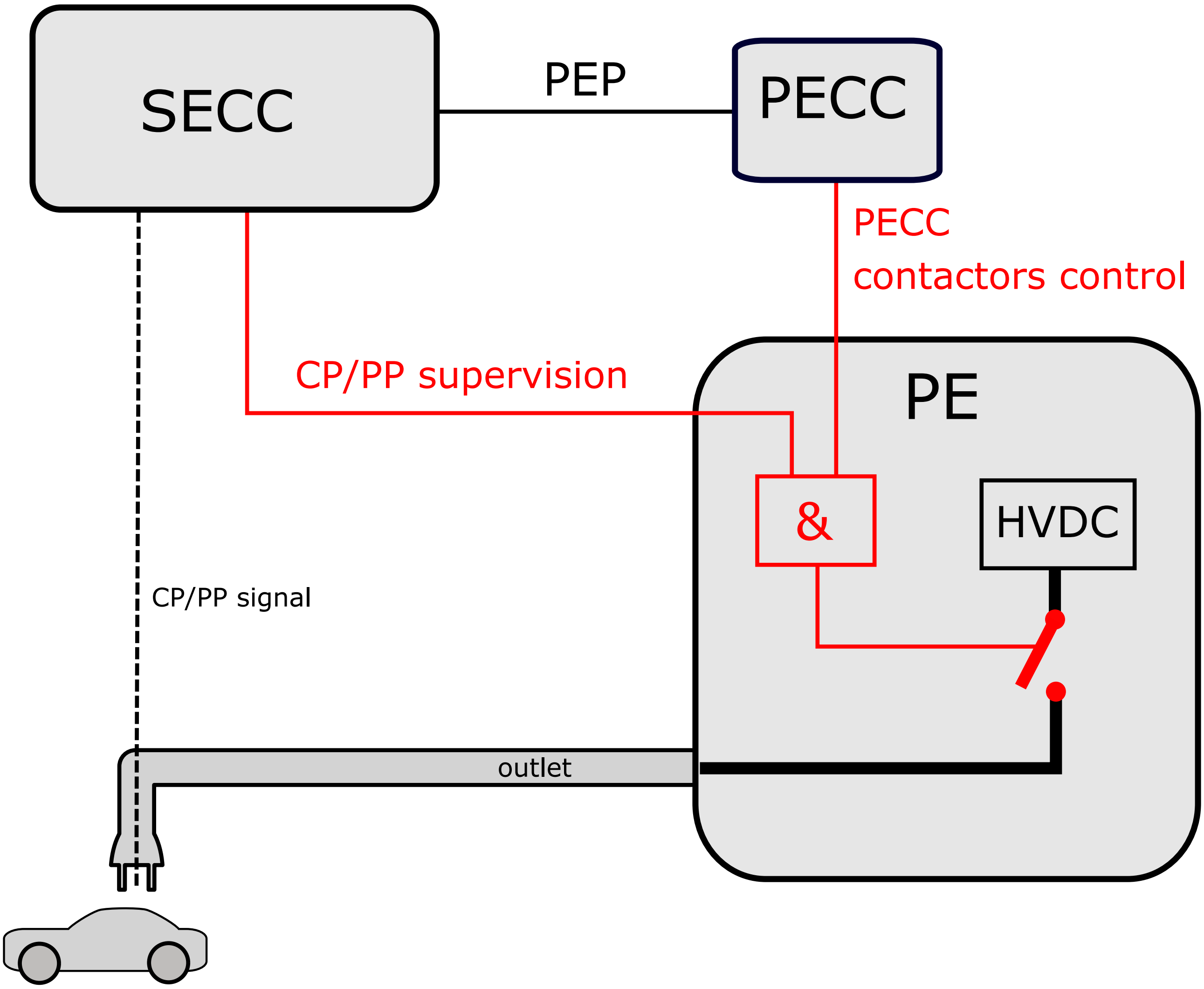
8 Appendix

8.1 IEC 61851 Control Pilot (CP) / Proximity Pin (PP) Supervision

The IEC 61851 and SAE J1772 standards impose strict safety requirements on the charging process and power supply monitoring. The charging process is controlled by the EV which sets a specific Control Pilot (CP) state. Six state categories exist: Ax, Bx, Cx, Dx, E and F. Energy transfer is allowed only in state categories Cx and Dx. In some cases (e.g. CCS Type 1) a PP supervision is also required to prevent energy transfer when the PP signal is not valid.

In order to implement this, the SECC provides a logical output called *CP/PP supervision*. This output controls the power electronics' ability to energize its outlet. Conceptually, a logical AND conjunction exists in the power electronics between the PECC control input and CP/PP supervision: The PE is able to close its contactors if and only if the CP/PP supervision allows it, i.e., the CP state category is Cx or Dx and PP signal is valid (if applicable).

If a contactorsStatus request could not be processed due to the CP/PP supervision, an error message with the errorCategory set to "internal" should be sent.



8.2 PEP JSON Schemas

8.2.1 request – configuration

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "configuration"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.2 request - cableCheck

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "cableCheck"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/cableCheckRequestPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    },
    "cableCheckRequestPayload": {
      "type": "object",
      "properties": {
        "voltage": {
          "$ref": "#/definitions/voltage"
        }
      },
      "required": [
        "voltage"
      ]
    },
    "voltage": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.3 request - targetValues

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "targetValues"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/targetValuesRequestPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    },
    "targetValuesRequestPayload": {
      "type": "object",
      "properties": {
        "targetVoltage": {
          "$ref": "#/definitions/targetVoltage"
        },
        "targetCurrent": {
          "$ref": "#/definitions/targetCurrent"
        },
        "batteryStateOfCharge": {
          "$ref": "#/definitions/batteryStateOfCharge"
        },
        "chargingState": {
          "$ref": "#/definitions/chargingState"
        }
      },
      "required": [
        "targetVoltage",
        "targetCurrent",
        "batteryStateOfCharge",
        "chargingState"
      ]
    },
    "targetVoltage": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "targetCurrent": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "batteryStateOfCharge": {
      "type": "number",
      "minimum": 0,
      "maximum": 100
    },
    "chargingState": {
      "$ref": "#/definitions/chargingStateType"
    },
    "chargingStateType": {
      "type": "string",
      "enum": [
        "standby",
        "preCharge",
        "charge",
        "postCharge"
      ]
    }
  }
}
```

8.2.4 request - contactorsStatus

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "contactorsStatus"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/contactorsStatusRequestPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    },
    "contactorsStatusRequestPayload": {
      "type": "object",
      "properties": {
        "contactorsStatus": {
          "$ref": "#/definitions/contactorStatusType"
        }
      },
      "required": [
        "contactorsStatus"
      ]
    },
    "contactorStatusType": {
      "type": "string",
      "enum": [
        "open",
        "closed"
      ]
    }
  }
}
```

8.2.5 request - getInput

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "getInput"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/getInputRequestPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
  "sequenceNumber": {
    "type": "integer",
    "minimum": 0,
    "maximum": 2147483647
  },
    "getInputRequestPayload": {
    "type" : "object",
     "properties": {
       "inputIdentifiers": {
         "$ref": "#/definitions/inputIdentifiers"
       }
     },
     "required": [
       "inputIdentifiers"
     ]
    },
    "inputIdentifiers": {
      "type": "array",
      "items": {
        "type" : "string"
      }
    }
  }
}
```

8.2.6 request - setOutput

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "setOutput"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/setOutputRequestPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
  "sequenceNumber": {
    "type": "integer",
    "minimum": 0,
    "maximum": 2147483647
  },
    "setOutputRequestPayload": {
     "type" : "object",
       "properties": {
        "outputValues": {
         "type": "object"
       }
     },
     "required": [
       "outputValues"
     ]
    }
  }
}
```

8.2.7 request - reset

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "reset"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.8 request - stopCharging

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "request"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "stopCharging"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.9 response - configuration

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "configuration"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/configurationResponsePayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    },
    "configurationResponsePayload": {
      "type": "object",
      "properties": {
        "firmwareVersion": {
          "type": "string"
        },
        "manufacturer": {
          "type": "string"
        },
        "limitVoltageMin": {
          "$ref": "#/definitions/limitVoltageMin"
        },
        "limitVoltageMax": {
          "$ref": "#/definitions/limitVoltageMax"
        },
        "limitCurrentMin": {
          "$ref": "#/definitions/limitCurrentMin"
        },
        "limitCurrentMax": {
          "$ref": "#/definitions/limitCurrentMax"
        },
        "limitPowerMin": {
          "$ref": "#/definitions/limitPowerMin"
        },
        "limitPowerMax": {
          "$ref": "#/definitions/limitPowerMax"
        },
        "limitDischargeCurrentMin": {
          "$ref": "#/definitions/limitDischargeCurrentMin"
        },
        "limitDischargeCurrentMax": {
          "$ref": "#/definitions/limitDischargeCurrentMax"
        },
        "limitDischargePowerMin": {
          "$ref": "#/definitions/limitDischargePowerMin"
        },
        "limitDischargePowerMax": {
          "$ref": "#/definitions/limitDischargePowerMax"
        },
        "floatValues": {
          "type": "boolean"
        }
      },
      "required": [
        "firmwareVersion",
        "manufacturer",
        "limitVoltageMin",
        "limitVoltageMax",
        "limitCurrentMin",
        "limitCurrentMax",
        "limitPowerMax"
      ]
    },
    "limitVoltageMin": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitVoltageMax": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitCurrentMin": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitCurrentMax": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitPowerMin": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitPowerMax": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "limitDischargeCurrentMin": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "limitDischargeCurrentMax": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "limitDischargePowerMin": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "limitDischargePowerMax": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    }
  }
}
```

8.2.10 response - cableCheck

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "cableCheck"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.11 response - targetValues

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "targetValues"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.12 response - contactorsStatus

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "contactorsStatus"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.13 response - getInput

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "getInput"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/getInputResponsePayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
  "sequenceNumber": {
    "type": "integer",
    "minimum": 0,
    "maximum": 2147483647
  },
    "getInputResponsePayload":
    {
      "type": "object"
    }
  }
}
```

8.2.14 response - setOuput

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "setOutput"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {}
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
  "sequenceNumber": {
    "type": "integer",
    "minimum": 0,
    "maximum": 2147483647
  }
  }
}
```

8.2.15 response - reset

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "reset"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.16 response - stopCharging

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "response"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "stopCharging"
      ]
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    }
  }
}
```

8.2.17 error - error

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "error"
      ]
    },
    "kind": {
      "$ref": "#/definitions/kindType"
    },
    "sequenceNumber": {
      "$ref": "#/definitions/sequenceNumber"
    },
    "payload": {
      "$ref": "#/definitions/errorMessagePayload"
    }
  },
  "required": [
    "type",
    "kind",
    "sequenceNumber",
    "payload"
  ],
  "definitions": {
    "kindType": {
      "type": "string",
      "enum": [
        "configuration",
        "cableCheck",
        "targetValues",
        "contactorsStatus",
        "error",
        "event",
        "status"
      ]
    },
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0,
      "maximum": 2147483647
    },
    "errorMessagePayload": {
      "type": "object",
      "properties": {
        "errorCategory": {
          "$ref": "#/definitions/errorCategory"
        },
        "errorDetails": {
          "type": "string"
        }
      },
      "required": [
        "errorCategory",
        "errorDetails"
      ]
    },
    "errorCategory": {
      "type": "string",
      "enum": [
        "format",
        "value",
        "inoperative",
        "internal",
        "generic"
      ]
    }
  }
}
```

8.2.18 info - event

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "info"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "event"
      ]
    },
    "payload": {
      "$ref": "#/definitions/eventInfoPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "payload"
  ],
  "definitions": {
    "eventInfoPayload": {
      "type": "object",
      "properties": {
        "eventDetails": {
          "type": "string"
        }
      },
      "required": [
        "eventDetails"
      ]
    }
  }
}
```

8.2.19 info - status

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "info"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "status"
      ]
    },
    "payload": {
      "$ref": "#/definitions/statusInfoPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "payload"
  ],
  "definitions": {
    "statusInfoPayload": {
      "type": "object",
      "properties": {
        "measuredVoltage": {
          "$ref": "#/definitions/voltageType"
        },
        "measuredCurrent": {
          "$ref": "#/definitions/currentType"
        },
        "drivenVoltage": {
          "$ref": "#/definitions/voltageType"
        },
        "drivenCurrent": {
          "$ref": "#/definitions/currentType"
        },
        "temperature": {
          "type": "number"
        },
        "contactorsStatus": {
          "$ref": "#/definitions/contactorsStatusType"
        },
        "isolationStatus": {
          "$ref": "#/definitions/isolationStatusType"
        },
        "operationalStatus": {
          "$ref": "#/definitions/operationalStatusType"
        }
      },
      "required": [
        "measuredVoltage",
        "measuredCurrent",
        "drivenVoltage",
        "drivenCurrent",
        "temperature",
        "contactorsStatus",
        "isolationStatus",
        "operationalStatus"
      ]
    },
    "voltageType": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "currentType": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "contactorsStatusType": {
      "type": "string",
      "enum": [
        "open",
        "closed"
      ]
    },
    "isolationStatusType": {
      "type": "string",
      "enum": [
        "valid",
        "invalid",
        "warning",
        "fault"
      ]
    },
    "operationalStatusType": {
      "type": "string",
      "enum": [
        "operative",
        "inoperative"
      ]
    }
  }
}
```

8.2.20 info - evConnectionState

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "info"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "evConnectionState"
      ]
    },
    "payload": {
      "$ref": "#/definitions/evConnectionStatePayload"
    }
  },
  "required": [
    "type",
    "kind",
    "payload"
  ],
  "definitions": {
    "evConnectionStatePayload": {
      "type": "object",
      "properties": {
        "evConnectionState": {
          "type": "string",
          "enum": [
            "disconnected",
            "connected",
            "energyTransferAllowed",
            "error"
          ]
        },
        "vehicleId": {
          "type": "string"
        }
      },
      "required": [
        "evConnectionState"
      ]
    }
  }
}
```

8.2.21 info - chargingSession

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "comment": "pep1.9",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "info"
      ]
    },
    "kind": {
      "type": "string",
      "enum": [
        "chargingSession"
      ]
    },
    "payload": {
      "$ref": "#/definitions/chargingSessionInfoPayload"
    }
  },
  "required": [
    "type",
    "kind",
    "payload"
  ],
  "definitions": {
    "chargingSessionInfoPayload": {
      "type": "object",
      "properties": {
        "chargingProfileMaxPowerLimitWatts": {
          "$ref": "#/definitions/chargingProfileMaxPowerLimitWatts"
        },
        "timeToFullSocSeconds": {
          "$ref": "#/definitions/timeToFullSocSeconds"
        },
        "evMinVoltageVolts": {
          "$ref": "#/definitions/evMinVoltageVolts"
        },
        "evMaxVoltageVolts": {
          "$ref": "#/definitions/evMaxVoltageVolts"
        },
        "evMinCurrentAmperes": {
          "$ref": "#/definitions/evMinCurrentAmperes"
        },
        "evMaxCurrentAmperes": {
          "$ref": "#/definitions/evMaxCurrentAmperes"
        },
        "evMinPowerWatts": {
          "$ref": "#/definitions/evMinPowerWatts"
        },
        "evMaxPowerWatts": {
          "$ref": "#/definitions/evMaxPowerWatts"
        },
        "evMinDischargeCurrentAmperes": {
          "$ref": "#/definitions/evMinDischargeCurrentAmperes"
        },
        "evMaxDischargeCurrentAmperes": {
          "$ref": "#/definitions/evMaxDischargeCurrentAmperes"
        },
        "evMinDischargePowerWatts": {
          "$ref": "#/definitions/evMinDischargePowerWatts"
        },
        "evMaxDischargePowerWatts": {
          "$ref": "#/definitions/evMaxDischargePowerWatts"
        },
        "chargeMode": {
          "$ref": "#/definitions/chargeMode"
        }
      }
    },
    "chargingProfileMaxPowerLimitWatts": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "timeToFullSocSeconds": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMinVoltageVolts": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMaxVoltageVolts": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMinCurrentAmperes": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMaxCurrentAmperes": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMinPowerWatts": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMaxPowerWatts": {
      "type": "number",
      "minimum": 0,
      "maximum": 2147483647
    },
    "evMinDischargeCurrentAmperes": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "evMaxDischargeCurrentAmperes": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "evMinDischargePowerWatts": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "evMaxDischargePowerWatts": {
      "type": "number",
      "minimum": -2147483647,
      "maximum": 0
    },
    "chargeMode": {
      "type": "string",
      "enum": [
        "scheduled",
        "dynamic",
        "dynamicBpt"
      ]
    }
  }
}
```