OCPP 1.6 Errata sheet

2025-04

Errata sheet: for OCPP version: 1.6 edition 2 and 1.6 FINAL.

Version History

VERSION	DATE	DESCRIPTION
2025-04	2025-04-30	Errata sheet release 2025-04.
v4.1 Draft	2019-11-15	Add new errata
v4.0 Release	2019-10-23	4th release of the errata sheet New errata are marked with v4.0.
v3.0 Release	2017-09-08	3th release of the errata sheet New errata are marked with v3.0.
v2.0 Release	2017-03-27	2nd release of the errata sheet Errata have been reordered to match the chronological order of the 1.6 specification.
v1.0 Release	2016-03-31	First release

1. Scope

This document contains errata on the OCPP 1.6 specification. Any errata added after v3.0 of this errata sheet is an errata on OCPP 1.6 edition 2, and is also applicable for OCPP 1.6 FINAL (the first edition) unless marked otherwise.

1.1. Terminology and Conventions

Bold: when needed to clarify differences, bold text might be used.

Since document version v3.0 errata are marked with a version number, indicating when an errata was added.

2. Major errata

Problems with the content/definition of the messages, class and enumerations of the protocol.

None known

3. Minor errata

Improvements to the descriptions on how the protocol (should) work.

3.1. Page 7, section: 2.2: More than IEC 15118 might be taken into account for Smart Charging.

The definition of "Composite Charging Schedule" states that "IEC 15118 limits might be taken into account", but also other limits might be taken into account.

Old text	Also IEC 15118 limits might be taken into account.
New text	Local Limits might be taken into account.

3.2. Page 10, section: 3.3: Feature Profiles should be normative

Added in errata sheet v4.0 The table with messages per feature profiles the columns have been mixed in edition 2.

Old text	**New text**
REMOTE TRIGGER	RESERVATION
RESERVATION	SMART CHARGING
SMART CHARGING	REMOTE TRIGGER

3.3. Page 10, section: 3.3: Feature Profiles should be normative

The Feature Profiles paragraph states that it is "informative", but it is "normative".

Old text	This section is **informative**.
New text	This section is **normative**.

3.4. Page 13, section: 3.5.1: Missing requirement, cache persists reset

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 16, section 3.4.1

In section 3.4.1 there is a requirement about the persistence of the cache, but it does not mention 'reset'

Old text	Cache values SHOULD be stored in non-volatile memory, and SHOULD be persisted across reboots and power outages.
New text	Cache values SHOULD be stored in non-volatile memory, and SHOULD be persisted across reboots **, resets** and power outages.

3.5. Page 15, section: 3.5.4: For unlock cable after invalid id in StartTransaction.conf, same identifier should be used.

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 18, section 3.4.4

The text explaining what a Charge Point should do when a StartTransaction.conf is received is not 100% clear that it should be the same identifier, not an identifier of the same owner.

Old text	it SHOULD keep the Charging Cable locked until the owner presents his identifier.
New text	it SHOULD keep the Charging Cable locked until the same identifier (or an identifier with the same ParentIdTag) is used to start the transaction (StartTransaction.req) is presented.

3.6. Page 18, section 3.5: Better description how to determine start/end of Energy Transfer Period

Added in errata sheet v3.0

Description how the Central System can determine start/end of Energy Transfer Period can be improved.

Old text	A Central System MAY deduce the start and end of an Energy Transfer Period from the MeterValues that are sent during the Transaction.

New text	A Central System MAY deduce the start and end of an Energy Transfer Period from: the MeterValues that are sent during the Transaction, the status notifications: Charging, SuspendedEV and/or SuspendedEVSE. etc. Central System implementations need to take into account factors such as: Some EVs don't go to state SuspendedEV: they might continue to trickle charge. Some Charge Point don't even have a electrical meter.

3.7. Page 18, section 3.12.2: Add description of stacking without duration

Added in errata sheet v3.0

When using stacking for Smart Charging: a high stack level without a duration will cause lower profiles to be never executed.

Add note at end of section 3.12.2

New text	NOTE: If you use Stacking without a duration, on the highest stack level, the Charge Point will never fall back to a lower stack level profile."

3.8. Page 20, section 3.13.1: Effect of updating or deleting TxDefaultProfile during a transaction not defined

Added in errata sheet v4.0

It was not clear that a new TxDefaultProfile not only applies to running transactions with a TxDefaultProfile, but also to running transactions without a ChargingProfile. Furthermore, removing a TxDefaultProfile will cause running transactions to continue without a TxDefaultProfile.

To clarify this best, two additional paragraphs are needed.

Page	Section	Additional text
20	3.13.1	When an new or updated TxDefaultProfile is received and a transaction is ongoing that is not using a ChargingProfile or using the current TxDefaultProfile, the transaction SHALL continue, but switch to using the new or updated TxDefaultProfile. When TxDefaultProfile is removed, then running transactions, that started with that profile, SHALL continue without a TxDefaultProfile.
54	5.16.3	When the ongoing transaction is not using a ChargingProfile or using the TxDefaultProfile, and a new or updated TxDefaultProfile is received, the transaction SHALL continue, but switch to using the new or updated TxDefaultProfile.

3.9. Page 29, section: 3.1.6: Missing advice to send meter register value if available

Added in errata sheet v4.0

When a Charge Point has a meter which allows the value of Meter Register (counter) to be read. It is seen as best to send this value instead of start every transaction at 0.

Additional text	If a Charge Point contains a Meter that has a register that can be read (instead of counting pulses or some other way of measuring the amount of energy etc.) It is RECOMMENDED to send this register value in every MeterValue send instead of starting at 0 for every transaction.

3.10. Page 33, new Chapter about time notations

In addition to "3.13 Time Zones" to following should have been added about time notations:

New text	**3.14 Time notations** Implementations MUST use ISO 8601 date time notation. Message receivers must be able to handle fractional seconds and time zone offsets (another implementation might use them). Message senders MAY save data usage by omitting insignificant fractions of seconds.

3.11. Page 35, section: 4.2: Boot Notification: Note on behaviour while not accepted by Central system

Old text	While not yet accepted by the Central System, the Charge Point may allow locally authorized transactions if it is configured to do so, as described in Local Authorization & Offline Behaviour. Parties who want to implement this behaviour must realize that it is uncertain if those transactions can ever be delivered to the Central System.
New text	A Charge Point Operator MAY choose to configure a Charge Point to accept transactions before the Charge Point is accepted by a Central System. Parties who want choose to implement this behavior should realize that it is uncertain if those transactions can ever be delivered to the Central System. After a restart (for instance due to a remote reset command, power outage, firmware update, software error etc.) the Charge Point MUST again contact the Central System and SHALL send a BootNotification request. If the Charge Point fails to receive a BootNotification.conf from the Central System, and has no in-built non-volatile real-time clock hardware that has been correctly preset, the Charge Point may not have a valid date / time setting, making it impossible to later determine the date / time of transactions. It might also be the case (e.g. due to configuration error) that the Central System indicates a status other than Accepted for an extended period of time, or indefinitely. It is usually advisable to deny all charging services at a Charge Point if the Charge Point has never before been Accepted by the Central System (using the current connection settings, URL, etc.) since users cannot be authenticated and running transactions could conflict with provisioning processes.

3.12. Page 37, section: 4.5: Relation between FirmwareStatusNotification.req and FirmwareUpdate.req is missing.

There is no description about the relation between FirmwareUpdate.req and FirmwareStatusNotification.req in the specification.

Additional text	The FirmwareStatusNotification.req PDUs SHALL be sent to keep the Central System updated with the status of the update process, started by the Central System with a FirmwareUpdate.req PDU.

3.13. Page 38, section: 4.7: Missing description of configuration keys for MeterValues

There are a couple of required configuration keys that have a huge influence of how MeterValues work. But there is no reference or description for them in 4.7 Meter Values:

- ClockAlignedDataInterval
- MeterValuesAlignedData
- MeterValuesAlignedDataMaxLength
- MeterValuesSampledData
- MeterValuesSampledDataMaxLength
- MeterValueSampleInterval
- StopTxnAlignedData
- StopTxnAlignedDataMaxLength
- StopTxnSampledData
- StopTxnSampledDataMaxLength

New Chapter to be added to the specification:

3.14: Metering Data

This section is normative.

Extensive metering data relating to charging sessions can be recorded and transmitted in different ways depending on its intended purpose. There are two obvious use cases (but the use of meter values is not limited to these two):

- Charging Session Meter Values
- Clock-Aligned Meter Values

Both types of meter readings MAY be reported in standalone MeterValues.req messages (during a transaction) and/or as part of the *transactionData* element of the StopTransaction.req PDU.

3.14.1 Charging Session Meter Values

Frequent (e.g. 1-5 minute interval) meter readings taken and transmitted (usually in "real time") to the Central System, to allow it to provide information updates to the EV user (who is usually not at the charge point), via web, app, SMS, etc., as to the progress of the charging session. In OCPP, this is called "sampled meter data", as the exact frequency and time of readings is not very significant, as long as it is "frequent enough". "Sampled meter data" can be configured with the following configuration keys:

- MeterValuesSampledData
- MeterValuesSampledDataMaxLength
- MeterValueSampleInterval
- StopTxnSampledData
- StopTxnSampledDataMaxLength

MeterValueSampleInterval is the time (in seconds) between sampling of metering (or other) data, intended to be transmitted by "MeterValues" PDUs. Samples are acquired and transmitted periodically at this interval from the start of the charging transaction.

A value of "0" (numeric zero), by convention, is to be interpreted to mean that no sampled data should be transmitted.

MeterValuesSampledData is a comma separated list that prescribes the set of measurands to be included in a MeterValues.req PDU, every MeterValueSampleInterval seconds. The maximum amount of elements in the MeterValuesSampledData list can be reported by the Charge Point via: **MeterValuesSampledDataMaxLength**

StopTxnSampledData is a comma separated list that prescribes the sampled measurands to be included in the TransactionData element of StopTransaction.req PDU, every MeterValueSampleInterval seconds from the start of the charging session. The maximum amount of elements in the StopTxnSampledData list can be reported by the Charge Point via: **StopTxnSampledDataMaxLength**

3.14.2 Clock-Aligned Meter Values

Grid Operator might require meter readings to be taken from fiscally certified energy meters, at specific Clock aligned times (usually every quarter hour, or half hour).

"Clock-Aligned Billing Data" can be configured with the following configuration keys:

- ClockAlignedDataInterval
- MeterValuesAlignedData
- MeterValuesAlignedDataMaxLength
- StopTxnAlignedData
- StopTxnAlignedDataMaxLength

ClockAlignedDataInterval is the size of the clock-aligned data interval (in seconds). This defines the set of evenly spaced meter data aggregation intervals per day, starting at 00:00:00 (midnight).

For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals.

A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.

MeterValuesAlignedData is a comma separated list that prescribes the set of measurands to be included in a MeterValues.req PDU, every ClockAlignedDataInterval seconds. The maximum amount of elements in the MeterValuesAlignedData list can be reported by the Charge Point via: **MeterValuesAlignedDataMaxLength**

StopTxnAlignedData is a comma separated list that prescribes the set of clock-aligned periodic measurands to be included in the TransactionData element of StopTransaction.req PDU for every ClockAlignedDataInterval of the charging session. The maximum amount of elements in the StopTxnAlignedData list can be reported by the Charge Point via: **StopTxnAlignedDataMaxLength**

3.14.3 Multiple Locations/Phases

When a Charge Point can measure the same measurand on multiple locations or phases, all possible locations and/or phases SHALL be reported when configured in one of the relevant configuration keys.

For example: A Charge Point capable of measuring *Current.Import* on *Inlet* (all 3 phases) (grid connection) and

Outlet (3 phases per connector on both its connectors). *Current.Import* is set in MeterValuesSampledData. MeterValueSampleInterval is set to 300 (seconds). Then the Charge Point should send:

- a MeterValue.req with: connectorId = 0; with 3 SampledValue elements, one per phase with location = Inlet.
- a MeterValue.req with: connectorId = 1; with 3 SampledValue elements, one per phase with location = Outlet.
- a MeterValue.req with: connectorId = 2; with 3 SampledValue elements, one per phase with location = Outlet.

3.14.4 Unsupported measurands

When a Central System sends a ChangeConfiguration.req to a Charge Point with one of the following configuration keys:

- MeterValuesAlignedData
- MeterValuesSampledData
- StopTxnAlignedData
- StopTxnSampledData

If the comma separated list contains one or more measurands that are not supported by this Charge Point, the Charge Point SHALL respond with: ChangeConfiguration.conf with: status = Rejected. No changes SHALL be made to the currently configuration.

Added in errata sheet v3.0

3.14.5 No metering data in a Stop Transaction

When the configuration keys: **StopTxnAlignedData** and **StopTxnSampledData** are set to an empty string, the Charge Point SHALL NOT put meter values in a StopTransaction.req PDU.

3.14. Page 31, section: 3.16.5: Missing description of Start and Stop MeterValues in StopTransaction.req

Added in errata sheet v4.0

Additional Section should have been added to 3.16:

3.16.6 Start and Stop MeterValues in StopTransaction.req

When the Charge Point is configured to provide metering data in the StopTransaction.req, the Charge Point is RECOMMENDED to always provide the start (Transaction.Begin) and stop (Transaction.End) values of every measurand configured.

If the Charge Point has to drop meter values because it is running out of memory, it is RECOMMENDED not to drop the start and stop values.

3.15. Page 36, section: 4.7: make it more explicit that MeterValues are required if they are configured

Added in errata sheet v4.0

The first paragraph of section 4.7 reads:

"A Charge Point MAY sample the electrical meter or other sensor/transducer hardware to provide extra information about its meter values. It is up to the Charge Point to decide when it will send meter values. This can be configured using the ChangeConfiguration.req message to data acquisition intervals and specify data to be acquired & reported."

The keyword "MAY" in this text suggests that it is optional to send MeterValues. However, when the value of the configuration variable MeterValueSampleInterval has been set to a value greater than zero, then it is mandatory to send MeterValues.

Additional text After first section	When the value of the configuration variable MeterValueSampleInterval and/or ClockAlignedDataInterval has been set to a value greater than zero, the Charge Point SHALL send MeterValues at the given interval(s).

3.16. Page 36, section: 4.7: Not 100% clear which transaction ID to use in a MeterValue (multiple transactions during interval)

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 38

It was not 100% clear which transactionId to put a MeterValue.req message for a Clock-Aligned meter value sample when during the last period between samples, the previous transaction has ended and a new transaction has started.

Additional text Bullet 2	The transactionId is always the ID of the Transaction that is ongoing (on that connector) at the moment the meter value sample is taken. When the TransactionId is set, only MeterValues related to that transaction can be reported. If the values are taken from the main energy meter, it is advised NOT to add a transactionId. When reporting Meter Values for connectorId 0 (the main energy meter) it is RECOMMENDED NOT to add a TransactionId.

3.17. Page 37, section: 4.8: Unclear how to handle: StartTransaction.conf with idTagInfo not Accepted.

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 40

It is not (clearly) defined what a Charge Point should do when it receives a StartTransaction.conf with authorization status other then Accepted, when it is online.

There is already text that describes what a Charge Point should do, but this text is now in the part explaining what to do when the Charge Point is offline. But because it is in that part, it is not clear that it is also they way a Charge Point should handle the same situation when it is online.

The following text should be moved from: Page 15, section: 3.5.4 (OCPP 1.6 FINAL Page 18, section 3.4.4) to: Page 37, section 4.8 (OCPP 1.6 FINAL Page 40)

"When the authorization status in the StartTransaction.conf is not Accepted, and the transaction is still ongoing, the Charge Point SHOULD:

- when StopTransactionOnInvalidId is set to true: stop the transaction normally as stated in Stop Transaction. The Reason field in the Stop Transaction request should be set to DeAuthorized. If the Charge Point has the possibility to lock the Charging Cable, it SHOULD keep the Charging Cable locked until the owner presents his identifier.
- when StopTransactionOnInvalidId is set to false: only stop energy delivery to the vehicle.

```
Note: In the case of an invalid identifier, an operator MAY choose to
charge the EV with a minimum
amount of energy so the EV is able to drive away. This amount is
controlled by the optional
configuration key: MaxEnergyOnInvalidId."
```

3.18. Page 37, section: 4.8: How to deliver Transaction related messages when no transactionId is known.

Added in errata sheet v4.0

This section mentions at the end, that failing to respond with a StartTransaction.conf will cause the Charge Point to retry several times. However, it is not specified how to deal with Transaction related messages for which no transactionId is known, when this happens.

At the end of section 4.8 add the following text:

Additional text	If the Charge Point was unable to deliver the StartTransaction.req despite repeated attempts, or if the Central System was unable to deliver the StartTransaction.conf response, then the Charge Point will not receive a transactionId. In that case, the Charge Point SHALL send any Transaction related messages for this transaction to the Central System with a transactionId = -1. The Central System SHALL respond as if these messages refer to a valid transactionId, so that the Charge Point is not blocked by this.

3.19. Page 38, section: 4.9: Not defined that StatusNotification should always be send.

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 40

The specification on page 38 states: "The following table depicts changes from a previous status (left column) to a new status (upper row) upon which a Charge Point MAY send a StatusNotification.req PDU to the Central System."

But the idea has always been that the Charge Point has to send a StatusNotification when the state changes.

Old text	a Charge Point **MAY** send a StatusNotification.req PDU to the Central System.
New text	a Charge Point **SHALL** send a StatusNotification.req PDU to the Central System, taking into account some minimum status duration.

3.20. Page 41, section: 4.9: Status transition from Preparing to Finishing (B6) is possible

Added in errata sheet v3.0

The original 1.6 does not allow a transition from Preparing to Finishing (B6). But there is a use case where this is possible:

A Charge Point with 2 connectors and 1 RFID reader. Driver 1 connects his charging cable. State goes to "Preparing" User does not authorize, times out. Then the Charge Point has to go to state: "Finishing". It should NOT go to "Available". Because if the charging cable remains plugged in and another EV driver swipes his RFID (before plugging in), that would start a transaction on the already plugged in cable. By going to "Finishing" this is prevented.

Table top page 41	Add **B6**
Table on page 42	B6: Timed out. Usage was initiated (e.g. insert plug, bay occupancy detection), but idTag not presented within timeout.

3.21. Page 42, section: 4.9 (B1): Missing reference to configuration key: "ConnectionTimeOut"

The configuration key: "ConnectionTimeOut" is never referenced in the specification, section: 4.9 needs a note explaining how the state: "Preparing" and the configuration key: "ConnectionTimeOut" work together.

Old text	Intended usage is ended (e.g. plug removed, bay no longer occupied, second presentation of idTag, time out on expected user action)
New text	Intended usage is ended (e.g. plug removed, bay no longer occupied, second presentation of idTag, time out **(configured by the configuration key: ConnectionTimeOut)** on expected user action)

3.22. Page 43, section: 4.9: Not defined that StatusNotification should always be send after BootNotification accepted.

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 45

There is no description of behaviour that is expected and is implemented by every known implementation of OCPP: After being accepted via BootNotification.conf(Accepted) the Charge Point has to report its current status.

Additional text	After the Central System accept a Charge Point by sending a BootNotification.conf with a status Accepted, the Charge Point SHALL send a StatusNotification.req PDU for connectorId 0 and all connectors with the current status.

3.23. Page 46, section: 5.4. Clear Cache when Cache is not implemented not defined.

Added in errata sheet v4.0

OCPP 1.6 FINAL Page 49

In OCPP 1.6, the Cache is not required, but the message: ClearCache.req is required to be implemented. OCPP does not define what the expected behaviour is.

Additional text	When the Authorization Cache is not implemented and the Charge Point receives a ClearCache.req message. The Charge Point SHALL response with ClearCache.conf with the status: Rejected.

3.24. Page 46, section 5.8: Missing requirement about chargingRateUnit in GetCompositeSchedule

Added in errata sheet v4.0

The following note must be added at the end of this section:

If an invalid value is used for chargingRateUnit in GetCompositeSchedule.req, e.g. when using a value for chargingRateUnit that is not 'A' or 'W', then Charge Point SHALL respond with RPC Framework CALLERROR: PropertyConstraintViolation (JSON) or SOAP Fault: Sender, ProtocolError (SOAP).

3.25. Page 47, section 5.5: Unclear how to match fields in a ClearChargingProfile request

Added in errata sheet v4.0

The following note must be added at the end of this section:

If no fields are specified in the ClearChargingProfile.req message, then Charge Point SHALL clear all ChargingProfiles. If one or more fields are specified in the ClearChargingProfile.req message, then Charge Point SHALL clear all ChargingProfiles that match (logical AND) all of the provided fields.

3.26. Page 63, section 6.13: Unclear how to match fields in a ClearChargingProfile request

Added in errata sheet v4.0

The following sentence in the description of ClearChargingProfile.req must be replaced with new text:

Old text	The Central System can use this message to clear (remove) either a specific charging profile (denoted by id) or a selection of charging profiles that match with the values of the optional connectorId, stackLevel and chargingProfilePurpose fields.

New text	The Central System can use this message to clear (remove) either a specific charging profile (denoted by id) or a selection of charging profiles that match (logical AND) with the values of the optional connectorId, stackLevel and chargingProfilePurpose fields. If no fields are provided, then all charging profiles will be cleared. If **id** is specified, then all other fields are ignored.

3.27. Page 50, section: 5.7: More than IEC 15118 might be taken into account for Smart Charging.

It is stated that "IEC 15118 limits might be taken into account", but also other limits might be taken into account.

Old text	Also IEC 15118 limits might be taken into account.
New text	Local Limits might be taken into account.

3.28. Page 51, section: 5.7: Get Composite Schedule: First sentences is not clear

In the description of Get Composite Schedule, it is not clear, what are the start and end points in time of the schedule that is to be sent.

Old text	Upon receipt of a GetCompositeSchedule.req, the Charge Point SHALL calculate the **scheduled time intervals up to the Duration is met and send them** to the central system.
New text	Upon receipt of a GetCompositeSchedule.req, the Charge Point SHALL calculate the **Composite Charging Schedule intervals, from the moment the request PDU is received: Time X, up to X + Duration, and send them in the GetCompositeSchedule.conf PDU** to the central system.

3.29. Page 51, section: 5.7: Get Composite Schedule: First sentences is not clear

Added in errata sheet v4.0

The description of Get Composite Schedule, can be clearified a bit more.

Old text	Upon receipt of a GetCompositeSchedule.req, the Charge Point SHALL calculate the **Composite Charging Schedule intervals, from the moment the request PDU is received: Time X, up to X + Duration, and send them in the GetCompositeSchedule.conf PDU** to the central system.
New text	Upon receipt of a GetCompositeSchedule.req, the Charge Point SHALL calculate the **scheduled time intervals from the moment of message receipt up to the Duration (in seconds) and send them** to the central system.

3.30. Page 51, section: 5.7: Improve use of connectorId '0' in GetCompositeSchedule

In the description of the use of connectorId '0' in GetCompositeSchedule it is not clear that it can also mean the Charge Point reports current instead of power.

Old text	If the ConnectorId in the request is set to '0', the Charge Point SHALL report the total expected **energy flow of the Charge Point for** the requested time period.
New text	If the ConnectorId in the request is set to '0', the Charge Point SHALL report the total expected **power or current the Charge Point expects to consume from the grid during** the requested time period.

3.31. Page 51, section: 5.9: Relation between GetDiagnostics.req and DiagnosticsStatusNotification.req is missing.

There is no description about the relation between GetDiagnostics.req and DiagnosticsStatusNotification.req in the specification.

The following diagram should replace the diagram in 5.9.

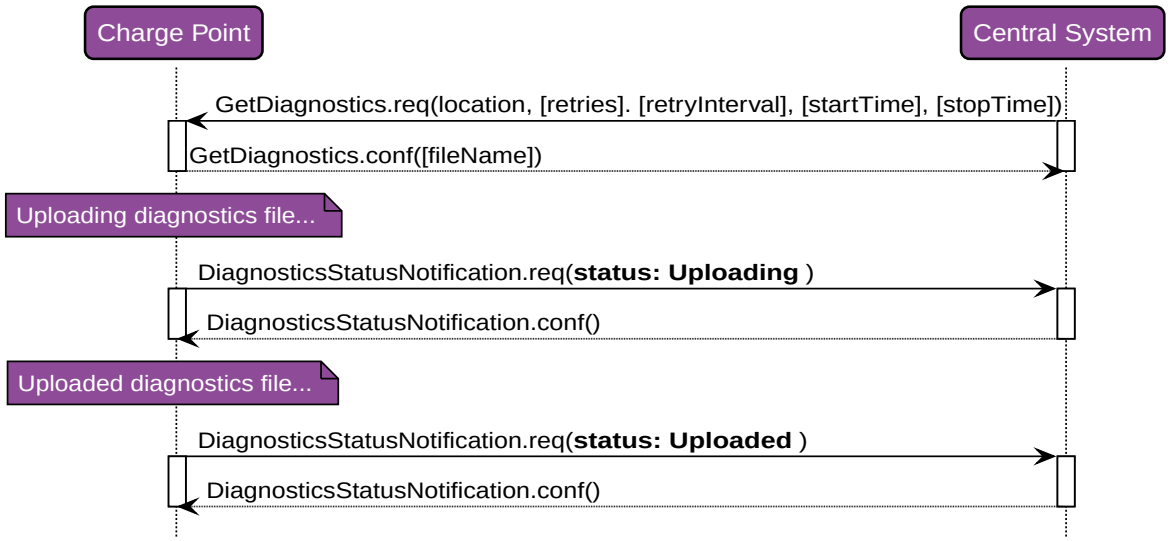

Figure 1. Sequence Diagram: get diagnostics

Additional text	During uploading of a diagnostics file, the Charge Point MUST send DiagnosticsStatusNotification.req PDUs to keep the Central System updated with the status of the upload process.

3.32. Page 52, section: 5.11. Remote Start Transaction: .conf status is accepted request

Old text	Central System can request a Charge Point to start a transaction by sending a RemoteStartTransaction.req. Upon receipt, the Charge Point SHALL reply with RemoteStartTransaction.conf and a status indicating whether it **is able** to start a transaction **or not**.
New text	Central System can request a Charge Point to start a transaction by sending a RemoteStartTransaction.req. Upon receipt, the Charge Point SHALL reply with RemoteStartTransaction.conf and a status indicating whether it **has accepted the request and will attempt** to start a transaction.

3.33. page 54, section: 5.12. Remote Stop Transaction: .conf status is accepted request