IMPLEMENTATION GUIDE

of Customer-Focused Key Performance Indicators for Electric Vehicle Charging

CHARGEX consortium

DECEMBER 2024

INL/RPT-24-77389

Argonne NATIONAL LABORATORY

INL Idaho National Laboratory

NREL Transforming ENERGY

Disclaimer

Authors

Mayuresh Savargaonkar[1,2], Michael Cabatingan[2], Kaleb Houck[2], Casey Quinn[2], Paden Rumsey[2], and John Smart[2]

Contributors

The authors gratefully acknowledge the invaluable contribution of industry participants in the ChargeX Consortium, including representatives from ABB E-mobility, Accenture, American Honda, AMPECO, Atlas Public Policy, BMW of North America, bp pulse, ChargeHub, ChargeMate, ChargePoint, Cool the Earth, Driivz, Electrify America, Enel X Way, EVBox, EVgo, FLO, Ford Motor Company, FreeWire Technologies, General Motors, Hubject, JD Power, KIGT, Lucid Motors, New York Power Authority, NovaCharge, Paren, Plug-In America, Qualcomm Inc., Red Inc. Communications, Rove, SAE International, Siemens, Tesla, University of California–Davis, University of Washington, VinFast US, and Xeal.

Acknowledgments

The National Charging Experience Consortium (ChargeX Consortium) is a collaborative effort between Argonne National Laboratory, Idaho National Laboratory, National Renewable Energy Laboratory, electric vehicle charging industry experts, consumer advocates, and other stakeholders. This document was produced by the ChargeX Consortium's Key Performance Indicators Task Force (KPI TF), formally known as "Working Group 1: Defining the Charging Experience."

The authors would like to acknowledge the Joint Office of Energy and Transportation (Joint Office) for supporting this work. The Joint Office and national laboratory staff would like to recognize the invaluable contribution of industry partners in providing the input summarized in this report.

[1] mayuresh@inl.gov
[2] Idaho National Laboratory

List of Acronyms

CCS	Combined Charging System
CPP	Customer Pain Point
CSO	Charging Station Operator, also referred to as a Charge Point Operator
EMSP	e-Mobility Service Provider
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
KPI	Key Performance Indicator
OCPI	Open Charge Point Interface
OCPP	Open Charge Point Protocol
WG	Working Group

Revision Log

Version Date	Revision	Author
12.31.2024	0	All authors and contributors
01.10.2025	1	Casey Quinn

Definitions

Charger. A device with one or more charging ports and connectors for charging electric vehicles (EVs). Also referred to as electric vehicle supply equipment (EVSE).

Charging network. A collection of chargers located on one or more properties that are connected via digital communications to manage the facilitation of payment, the facilitation of electrical charging, and any other related data requests.

Charging port. The system within a charger that charges an EV. A charging port may have multiple connectors, but it can provide power to charge only one EV through one connector at a time.

Charging station. The area in the immediate vicinity of a group of chargers that includes the chargers, supporting equipment, parking areas adjacent to the chargers, and lanes for vehicle ingress and egress. A charging station could comprise only part of the property on which it is located.

Charging station operator. The entity that owns the chargers and supporting equipment and facilities at one or more charging stations. Although this entity may delegate responsibility for certain aspects of charging station operation and maintenance to subcontractors, this entity retains responsibility for operation and maintenance of the chargers, supporting equipment, and facilities. In some cases, the charging station operator and the charging network provider are the same entity.

Connector. The device that attaches an EV to a charging port in order to transfer electricity.

Table of Contents

List of Tables

1. Introduction

Electric vehicle (EV) sales account for a rapidly growing portion of the light-duty vehicle market. However, the growth of EV adoption is inherently tied to the reliability and usability of public EV charging. Today, customers of public charging stations, EV drivers, frequently encounter such problems as lengthy wait times, trouble initiating charging sessions, and slow charging speeds. It is crucial for the EV-charging industry to understand and address these issues to improve the customer experience and ensure a continuing upward trend in EV adoption.

1.1 Measurement Is Required for Improvement

To systematically improve the public charging experience, EV-charging-industry stakeholders need to define and measure it precisely. Many stakeholders currently measure aspects of the charging experience, but they typically employ metrics that are either operational in nature, such as charger uptime and mean time between failures, or composite customer-satisfaction indices. To improve the customer experience most effectively, the industry needs metrics that define the charging experience from the perspective of the customer, not business operations. Furthermore, industry practitioners need granular metrics to know what specific aspects of the charging experience need improvement. This report defines such customer-focused metrics, called key performance indicators (KPIs).

1.2 Shared Responsibility

Although charging-station operators (CSOs) are often perceived as bearing the responsibility for the charging experience, numerous other stakeholders share this responsibility, including EV manufacturers, charger manufacturers, and electric-mobility service providers (EMSPs; i.e., third-party map and payment app developers). To effectively improve the charging experience, this ecosystem of interdependent companies must uniformly adopt common, customer-focused KPIs and measurement methods to ensure a common understanding. Additionally, no single stakeholder currently generates or has access to all the data necessary to provide full visibility of the charging experience.[3] Cross-industry coordination and innovation are required to achieve this.

For these reasons, the ChargeX Consortium established Working Group 1: Defining the Charging Experience. This group includes representatives from CSOs, charger manufacturers, EV manufacturers, EMSPs, field service providers, national laboratories, consumer-advocacy and non-profit organizations, and academia who specialize in EV charging customer research.

2. Purpose of this Report

This report describes how individual and unique messages sent through Open Charge Point Protocol (OCPP) sessions and/or transactions are used to calculate the interim set of KPIs

[3] The KPIs defined here are not intended for use in a way that isolates responsibility with any single stakeholder. However, it is foreseeable that funding programs will need KPIs for funding recipients, and the definitions herein may be useful for that purpose. In that case, these definitions may require modification and/or exemptions in order to accurately measure performance of any single stakeholder.

established by ChargeX Consortium Working Group 1: Defining the Charging Experience in the report entitled, "Customer-Focused Key Performance Indicators for Electric Vehicle Charging."[4]

2.1 Interim Set of KPIs to Provide Near-Term Benefits

The interim set of KPIs are those that can be calculated by individual companies using data currently generated and communicated via OCPP. These KPIs provide a limited view of the charging experience, but they are implementable in the near-term, thus allowing them to provide benefits to address challenges faced today.

2.2 Intended Audience

The intended audience for this report is industry practitioners. The report does not provide policy recommendations. Instead, the intent is to mature both individual industry stakeholders' capabilities and the industry's collective capability to improve the public charging experience by establishing uniform methods for measuring the experience.

3. Implementation of the Interim Key Performance Indicators[5]

Two sets of KPIs were needed to address the customer pain points: (1) an interim set for near-term assessments; and (2) an ideal set to provide a full view of key aspects of the charging experience. These two sets are needed as the interim KPIs are not all encompassing of the charging experience, but can be calculated today as well as in the near future. The ideal KPIs provide more insights but will require significant efforts by the industry to implement. This document only addresses the interim set of KPIs that consist of the following:

1. Charge start success (%)
2. Charge start time (seconds)
3. Charge end success (%)
4. Session success (%).

Figure 1 shows the relationship of the interim set of KPIs to the charging experience. The remainder of the report provides a detailed guide for the implementation of these interim KPIs.

[4] INL/RPT-24-77388.

[5] KPIs are calculated using the summation of relevant sessions and transactions for a given charge station and will never exceed 100% or drop below 0%.

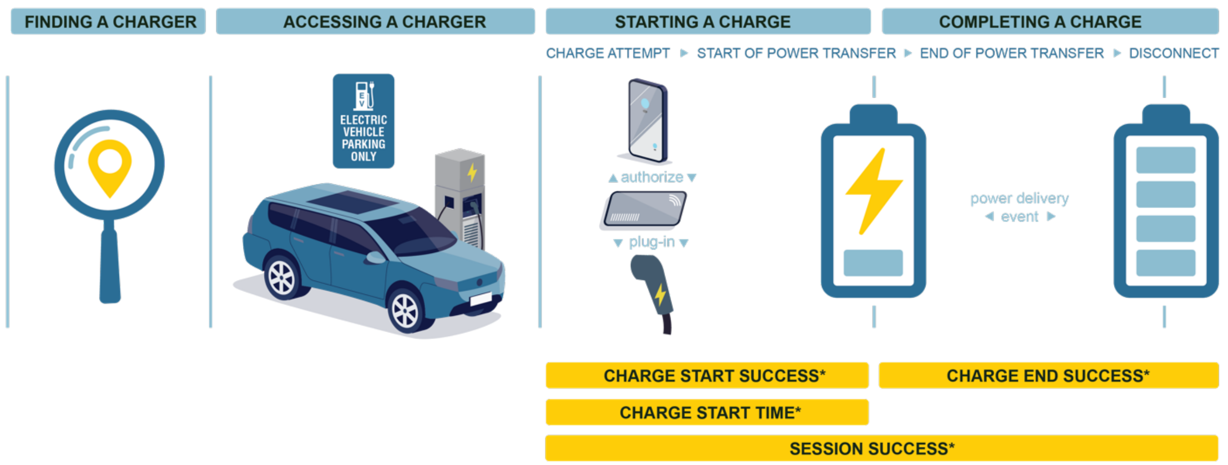

Figure 1: Interim set of KPIs.

3.1 Charge Start Success

Charge Start Success defines the percent of charge attempts[6] that result in an EVSE starting to deliver power to an EV.

This KPI measures the fraction of all charge attempts made by all customers over a period of time that were successful, meaning all steps required to start the delivery of power to the EV occurred without requiring the customer to repeat actions or otherwise intervene (e.g., payment authorization, EV or EV driver authentication, establishing communication between the EV and EVSE, etc.).

Charge Start Success is measured as a percentage and applies to one or more charging ports at one or more charging stations. For the purposes of this work, a "Plug-In Attempt" is defined as an attempt made by an EV driver to connect an EV to an EVSE port with the intent of initializing and successfully starting a charge. In OCPP 2.0.1, this can be identified when a Charging Station Management System (CSMS) is notified of a transaction start using *TransactionEventRequest message* 1.60.1 in OCPP 2.0.1. Several *triggerReasons* are defined in Section 3.82 in OCPP 2.0.1 for a transaction start identification, including *CablePluggedIn.* In OCPP 1.6J, a plug-in attempt can be identified when a CSMS is notified of a status change using *StatusNotification.req message* 4.9 and 6.47 in OCPP 1.6J. *ChargePointStatus* is an enumeration that is defined in Section 7.7 in OCPP 1.6J. It can be used for a transaction start identification by looking for the *Preparing* value, which indicates a cable plug-in.[7]

Similarly, "power delivery" can be determined by OCPP 2.0.1 with a *TransactionEventRequest (triggerReason ="ChargingStateChanged", transactionInfo:chargingState ="Charging")*

---

[6] *Charge attempts* are defined as either: (a) an EV driver connects the EV to an EVSE; or (b) the EV driver begins authorization (i.e., valid credentials and/or payment has been provided) for a charging session.

[7] Figure 5 in "Open Charge Point Protocol 1.6 edition 2 FINAL, 2017-09-28" provides a diagram that explains the various steps and terminology used by OCPP for the charging process.

message. In OCPP 1.6J, the equivalent is reflected using a *StatusNotification.req* message with *StatusNotification.req(status= "Charging").*[8,9,10]

Finally, an "attempt to authorize" may be defined as the intent of the EV driver to pay for the charging session.[11] There are two nominal use-cases: (1) authorize after plug-in; and (2) authorize before plug-in.[12] More details are presented below.

3.1.1 Authorize After Plug-In Using OCPP 2.0.1

In this scenario, the EV/EV driver authorizes the transaction after plugging in. Per OCPP 2.0.1 Section C.2.1, there are eight ways in which a transaction may be authorized. Of these eight, only two require plug-in before authorization is required:

- C07 – Authorization using Contract Certificates
- C08 – Authorization at EVSE using ISO 15118 External Identification Means (EIM).

In both scenarios, for every plug-in attempt, the *TransactionEventRequest* message can be tracked (Section 1.60.1 in OCPP 2.0.1), so the Charge Start Success may be calculated using Eq. 1:

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{TransactionEventRequest(triggerReason = "CablePluggedIn", eventType = "Started")}}\right) \text{x}100 \tag{1}$$

where,[13,14]

[8] Successful start of power delivery may be seen as a state transition into State C or D with a nominal voltage of 6VDC per SAE J1772 2024 Tables 1 and 2.

[9] Per ISO 15118, the EV switches to State C after sending the *PowerDeliveryReq (ChargeProgress = "Start")* while AC charging (Fig. 105-106 in ISO 15118-2) and after receiving the *ChargeParameterDiscoveryRes (ResponseCode = "OK", EVSEProcessing = "Finished")* (Fig. 107-108 in ISO 15118-2) while DC charging.

[10] To include failures that may occur during CableCheck and PreCharge in DC charging, successful start of power delivery is considered to be the moment the Supply Equipment Communication Controller (SECC) receives a *PowerDeliveryReq (ChargeProgress = "Start")* and responds with a *PowerDeliveryRes (ResponseCode = "OK").*

[11] When parties implementing this KPI agree to exemptions, exempt Plug-In Attempts (or Attempts to Authorize) may be subtracted from the denominator in this KPI. In that case, each exemption should be tracked as a Plug-In Attempt (or Authorize Attempt) log paired with its eventual corresponding failure log that indicates an agreed exempt failure reason.

[12] In the future, OCPP supports *ParkingBayOccupied* and when this feature is more widely implemented, it is suggested to consider adding all *TxStartPoints* in Section 2.6.4 of OCPP 2.0.1 as start points.

[13] The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

[14] There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized").*

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \text{ if } \sum \text{TransactionEventRequest}\begin{pmatrix} \text{triggerReason} = \text{"ChargingStateChanged"}, \\ \text{transactionInfo: chargingState} = \text{"Charging"} \end{pmatrix} \geq 1 \\ 0 \text{ otherwise} \end{cases}$$

3.1.2 Authorize After Plug-In Using OCPP 1.6J

In this scenario, the EV/EV driver authorizes the transaction after being asked to plug-in.

Alternatively, OCPP 1.6J can be used to calculate the Charge Start Success. Per OCPP 1.6J, there are two ways in which a transaction may be authorized:

- Operations initiated by Charge Point (Section 4 in OCPP 1.6J)
- Operations initiated by Central System (Section 5 in OCPP 1.6J).

In both scenarios, for every plug-in attempt, tracking both the S*tatusNotification.req* (Section 6.47 in OCPP 1.6J) and the *StartTransaction.conf* (Section 6.46 in OCPP 1.6J) so that the Charge Start Success may be calculated as:

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{StatusNotification.req(status} = \text{"Preparing")}}\right) \text{x100} \quad (2)$$

where, [15,16]

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \text{ if } \sum \text{StatusNotification.req(status} = \text{"Charging")} \mid \text{StartTransaction.conf(idTagInfo: status} = \text{"Accepted")} \geq 1 \\ 0 \text{ otherwise} \end{cases}$$

3.1.3 Authorize Before Plug-In Using OCPP 2.0.1

In this scenario, the EV/EV driver authorizes the transaction before being asked to plug-in or the connector is unlocked. This is particularly challenging since the EV is not connected and no EV-EVSE communication is established. Per OCPP 2.0.1 Section C.2.1, there are eight ways in which a transaction may be authorized. Of these eight, only six are applicable in this scenario, as shown in Table *1*.

[15] The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

[16] There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized").*

Table 1 outlines the message request-response pairs that may be tracked to calculate Charge Start Success for each possible authorization method in each transaction.

Table 1. Relevant request-response pairs for tracking Charge Start Success for OCPP 2.0.1.

Scenario	Request-Response Pair	Comments
C01 – EV Driver Authorization using RFID	TransactionEventRequest - AuthorizeResponse	Follow Eq. 3.
C02 – Authorization using a start button	-	The CSMS cannot reject the transaction. CSS = 100%.
C03 – Authorization using credit/debit card	Multiple	Follow Eq. 3 or Eq. 4 based on individual workflows.
C04 – Authorization using PIN-code	TransactionEventRequest - AuthorizeResponse	Follow Eq. 3.
C05 – Authorization for CSMS initiated transactions	TransactionEventRequest - RequestStartTransactionResponse	Follow Eq. 4.
C06 – Authorization using local id type	AuthorizeRequest - AuthorizeResponse	Follow Eq. 3.

Based on the request-response pairs given in Table 1, *Charge Start Success* may be calculated as:

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{AuthorizeResponse(idTokenInfo: status = "Accepted")} + \sum \text{AuthorizeResponse(idTokenInfo: status} \neq \text{"Accepted")}}\right) \text{x } 100 \quad (3)$$

Or

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{RequestStartTransactionResponse(status = "Accepted")} + \sum \text{RequestStartTransactionResponse(status = "Rejected")}}\right) \text{x } 100 \quad (4)$$

where, [17,18,19]

[17] The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

[18] There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized")*.

[19] Currently, it is not possible to track all failed attempts that are directly made by the user using a payment terminal on the charger for which no OCPP message(s) are generated. Such failures may include authentication failures that may occur on the payment system due to several factors including Internet connectivity failures, timeouts, bad card

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \; if \; \sum \text{TransactionEventRequest}\begin{pmatrix} \text{triggerReason} = \text{"ChargingStateChanged"}, \\ \text{transactionInfo: chargingState} = \text{"Charging"} \end{pmatrix} \geq 1 \\ 0 \text{ otherwise} \end{cases}$$

In some cases, it may be possible the transaction is "authorized" using the Local Authorization List (or any authorization cache). In such cases, it is possible the *AuthorizeRequest-Response* pair in Eq. 3 is not encountered. Instead, a *TransactionEventRequest-Response* pair will be seen. In such cases, the use of Eq. 5 is recommended to calculate *Charge Start Success*:

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\begin{matrix} \sum \text{TransactionEventResponse(idTokenInfo: status} = \text{"Accepted")} + \\ \sum \text{TransactionEventResponse(idTokenInfo: status} \neq \text{"Accepted")} \end{matrix}} \right) \text{x } 100 \quad (5)$$

3.1.4 Authorize Before Plug-In Using OCPP 1.6J

In this use-case, the EV/EV driver authorizes the transaction before being asked to plug-in, the connector is unlocked, and a session begins. This is particularly challenging since the EV is not connected and no EV-EVSE communication is established. Per OCPP 1.6J, there are two ways in which a transaction may be authorized:

- Operations Initiated by Charge Point (Section 4 in OCPP 1.6J)
- Operations Initiated by Central System (Section 5 in OCPP 1.6J).

Table 2 outlines the message request-response pairs that may be tracked to calculate Charge Start Success for each possible authorization method in each transaction.

Based on the request-response pairs given in Table 2, Charge Start Success may be calculated as the ratio of receipt of number of *StatusNotification.req(status = "Charging")* to the number of authorization requests received given that *StartTransaction.conf(idTagInfo:status = "Accepted")* was received.

readers, etc. Although these failures are indicated to the user on the payment terminal, the standard offers limited insights into such events and are deemed out-of-scope for the purpose of this document. It is recommended that the relevant standard bodies investigate these failures in more detail and address this gap in future editions of their standards.

Table 2. Relevant request-response pairs for tracking Charge Start Success for OCPP 1.6J.

Scenario	Request-Response Pair	Comments
EV Driver Authorization using RFID	StatusNotification.req – Authorize.conf	Follow Eq. 6.
Authorization using a start button	-	The CSMS cannot reject the transaction. CSS = 100%.
Authorization using a credit/debit card	Multiple	Follow Eq. 6 or Eq. 7 based on individual workflows.
Authorization using a PIN-code	StatusNotification.req - Authorize.conf	Follow Eq. 6.
Authorization for CSMS initiated transactions	StatusNotification.req – RemoteStartTransaction.conf	Follow Eq. 7.
Authorization using local authorization or cache	StatusNotification.req - Authorize.conf	Follow Eq. 6.

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{Authorize.conf(idTagInfo: status = "Accepted")} + \sum \text{Authorize.conf(idTagInfo: status} \neq \text{"Accepted")}}\right) \text{x } 100 \quad (6)$$

Or

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{RemoteStartTransaction.conf(status = "Accepted")} + \sum \text{RemoteStartTransaction.conf(status = "Rejected")}}\right) \text{x } 100 \quad (7)$$

where, [20,21,22]

[20] The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

[21] There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized").*

[22] Currently, it is not possible to track all failed attempts that are directly made by the user using a payment terminal on the charger for which no OCPP message(s) are generated. Such failures may include authentication failures that may occur on the payment system due to several factors including Internet connectivity failures, timeouts, bad card readers, etc. Although these failures are indicated to the user on the payment terminal, the standard offers limited insights into such events and are deemed out-of-scope for the purpose of this document. It is recommended that the relevant standard bodies investigate these failures in more detail and address this gap in future editions of their standards.

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \text{ if } \sum \text{StatusNotification.req(status = "Charging")} \mid \text{StartTransaction.conf(idTagInfo: status = "Accepted")} \geq 1 \\ 0 \text{ otherwise} \end{cases}$$

In some cases, it may be possible the transaction is "authorized" using the Local Authorization List (or any authorization cache). In such cases, it is possible the *Authorize.req- conf* pair in Eq. 6 is not encountered. Instead, a *StartTransaction.req-conf* pair will be seen. In such cases, the use of Eq. 8 is recommended to calculate *Charge Start Success*:

$$\text{Charge Start Success} = \left(\frac{\sum \text{PowerDeliveryAttempts}}{\sum \text{StartTransaction.conf(idTagInfo: status = "Accepted")} + \sum \text{StartTransaction.conf(idTagInfo: status} \neq \text{"Accepted")}} \right) \text{x } 100 \quad (8)$$

3.2 Charge Start Time

Charge Start Time can be defined as the time required for an EVSE to begin delivering power after a charge attempt is initiated.

This KPI measures how long it took to start delivering power to the EV from the time the customer initiates a charge attempt. This KPI includes customer dwell time (e.g., the time it takes the customer to find a credit card or radio frequency identification [RFID] card, open the charge door on the EV, and plug-in the charger [for chargers that require authorization first], etc.)

Charge Start Time is measured in seconds and applies to one or more charging ports at one or more charging stations. It is recommended that charge start time only be calculated for sessions that are successfully authorized and begin delivering power. This can be tracked using respective timestamps sent over every unique OCPP session (i) such that,

$$\text{Charge Start Time} = \left(t_{power,i} - t_{attempt,i}\right) \quad (9)$$

where,

t_{power} is defined as timestamp when power delivery begins and $t_{attempt}$ is defined as the timestamp when a charge attempt is made. Note the definitions of t_{power} and $t_{attempt}$ are relative to methods of charge attempts as defined in Section 3.1 in Eqs. 2 – 8. The median (50th percentile), 10th, 25th, 75th, and 90th percentiles should be calculated for each reporting period.

3.2.1 Authorization After Plug-In using OCPP 2.0.1

For authorization after plug-in using OCPP 2.0.1, t_{power} will be the timestamp when $\mathit{TransactionEventRequest}\left(\begin{matrix}\mathit{triggerReason} = \mathit{"ChargingStateChanged"}, \\ \mathit{transactionInfo{:}chargingState} = \mathit{"Charging"}\end{matrix}\right)$ is received, while $t_{attempt}$ will be the timestamp when

TransactionEventRequest(triggerReason ="CablePluggedIn", eventType ="Started") is received for this particular transaction.

3.2.2 Authorization After Plug-In using OCPP 1.6J

For authorization after plug-in using OCPP 1.6J, t_{power} will be the timestamp when *StatusNotification.req(status =”Charging”)| StartTransaction.conf(idTagInfo:status = "Accepted")* is received, while $t_{attempt}$ will be the timestamp when *StatusNotification.req(status ="Preparing")* is received for this particular transaction.

3.2.3 Authorization Before Plug-In using OCPP 2.0.1

For authorization after plug-in using OCPP 2.0.1, t_{power} will be the timestamp when *TransactionEventRequest (triggerReason ="ChargingStateChanged", transactionInfo:chargingState ="Charging")* is received and $t_{attempt}$ will be the timestamp when *AuthorizeResponse(idTokenInfo:status ="Accepted")* or *RequestStartTransactionResponse(status ="Accepted")* or *TransactionEventResponse(idTokenInfo:status ="Accepted")* is received for this particular transaction.

3.2.4 Authorization Before Plug-In using OCPP 1.6J

For authorization before plug-in using OCPP 1.6, t_{power} will be the timestamp when *StatusNotification.req(status ="Charging")| StartTransaction.conf(idTagInfo:status = "Accepted")* is received and $t_{attempt}$ will be the timestamp when *Authorize.conf(idTagInfo:status ="Accepted")* or *RemoteStartTransaction.conf(status = "Accepted")* or *StartTransaction.conf(idTagInfo:status = "Accepted")* is received for this particular transaction.

3.3 Charge End Success

Charge End Success measures the fraction of successful charge attempts made by all customers over a period of time resulting in: (a) the charging session completing without any errors that terminated or suspended charging unexpectedly; and (b) the customer being able to unplug without manual intervention to unlock the connector from the vehicle.

Charge End Success is measured as a percentage and applies to one or more charging ports at one or more charging stations.

3.3.1 Charge End Success using OCPP 2.0.1

Per OCPP 2.0.1, a successful session for a plug-in attempt may be defined as the receipt of *TransactionEventRequest (transactionInfo: stoppedReason = ”EnergyLimitReached” or “SOCLimitReached” or ”Local” or “Remote” or “EVDisconnected” or “StoppedByEV” or “LocalOutofCredit” or “TimeLimitReached”)* given that it is followed by the receipt of *StatusNotificationRequest(connectorStatus ≠ ”Occupied”).*

Charge End Success may be defined as the ratio of number of *TransactionEventRequest (transactionInfo:stoppedReason = ”EnergyLimitReached” or “SOCLimitReached” or ”Local”* *or "Remote" or "EVDisconnected" or "StoppedByEV" or "LocalOutofCredit" or "TimeLimitReached")* to the *PowerDeliveryAttempts* where, [23,24,25]

$$\text{Charge End Success} = \left(\frac{\begin{array}{c}\sum \text{TransactionEventRequest(transactionInfo: stoppedReason = "EnergyLimitReached"} \\ \text{or SOCLimitReachedor Local" or "Remote or "EVDisconnected"} \\ \text{or "StoppedByEV" or "LocalOutofCredit"} \\ \text{or "TimeLimitReached")}\end{array}}{\sum \text{PowerDeliveryAttempts}} \right) \text{x100} \qquad (10)$$

where,

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \; if \; \sum \text{TransactionEventRequest} \begin{pmatrix} \text{triggerReason = "ChargingStateChanged",} \\ \text{transactionInfo: chargingState = "Charging"} \end{pmatrix} \geq 1 \\ \text{0 otherwise} \end{cases}$$

A transaction can be reported as completed according to the above equations but may still have encountered errors during the charging session. It is recommended to see if any of these are 'active permanent faults' that resulted in a failed session. Some of these are defined by SAE J1772 JAN2024 and Minimum Required Error Codes (MREC).[26]

3.3.2 Charge End Success using OCPP 1.6J

Per OCPP 1.6J, a successful session for every plug-in attempt may be defined as the receipt of *StopTransaction.req (reason = "Remote" or "Local" or "EVDisconnected" or "UnlockCommand")*, given it is followed by the receipt of *StatusNotification.req (status≠ "Occupied")*. OCPP 1.6J does not contain "EnergyLimitReached" or "SOCLimitReached" in this list of reasons. It is recommended that "Local" is triggered instead in these instances.

[23] The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

[24] There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized").*

[25] Situations where the "user interactions" on an EV lead to a non-emergency or error shutdowns may be included in the "Local" stopped reason. Similarly, "user interactions" that lead to emergency or erroneous shutdowns cannot be calculated as successful charge sessions including the cases where the user pushes the emergency shutdown button or the S3 switch has been activated since they would result in an Emergency and/or Error Shutdown per Table F3 in SAE J1772 and per Table 3.

[26] https://inl.gov/chargex/mrec/

Charge End Success may be defined as the ratio of number of *StopTransaction.req (reason = "Remote" or "Local" or "EVDisconnected" or "UnlockCommand")* to the *PowerDeliveryAttempts* where, [27,28,29]

$$\text{Charge End Success} = \left(\frac{\sum \text{StopTransaction.req(reason = Local" or Remote or "EVDisconnected" or "UnlockCommand")}}{\sum \text{PowerDeliveryAttempts}}\right) x100 \tag{11}$$

where,

$$\text{PowerDeliveryAttempts} = \begin{cases} 1 \text{ if } \sum \text{StatusNotification.req(status = "Charging")} \,|\, \text{StartTransaction.conf(idTagInfo: status = "Accepted")} \geq 1 \\ 0 \text{ otherwise} \end{cases}$$

Table 3. List of errors that may result in a failed session.

Error	Source
Cable check errors	J1772
Insulation monitoring during energy transfer	J1772
Protection against overvoltage at the DC output between DC+ and DC-	J1772
Overtemperature handling	J1772
Check of the plausibility of values provided by the thermal sensing	J1772
Short-circuit before energy transfer	J1772
Maximum voltage between DC output live parts and productive conductor with a single fault	J1772
Loss of electrical continuity of control pilot conductor	J1772
Loss of electrical continuity of proximity detection conductor	J1772
Overcurrent protection	J1772
Control circuit supply integrity	J1772
Loss of electrical continuity of the protective conductor	J1772
Short-circuit protection	J1772
If error shutdown does not work properly	J1772
CX001 – ConnectorLockFailure	MREC
CX002 – GroundFailure	MREC
CX003 – HighTemperature	MREC
CX004 – OverCurrentFailure	MREC
CX005 – OverVoltage	MREC

27 The *Power Deliver Attempts used to calculate Charge State Success* variable is a binary variable tracked using *TransactionEventRequest (eventType = "Started")*. This is required since there may be instances when multiple *TransactionEventRequest (transactionInfo: chargingState = "Charging")* are received when a transaction is paused and resumed. In such cases, the cable remains connected and that requires *PowerDeliveryAttempts* to be a binary variable to limit the Charge Start Success to be a maximum of 100% per plug-in. The current calculation of this metric has the possibility to report Charge Start Success to be 100% even if it fails to deliver power after resuming a session. However, this will be addressed in future revisions.

28 There may be cases where the EV is plugged-in and uses plug and charge while a minimal amount of energy is being delivered while authorization is under process. In such cases if authorization fails, the *PowerDeliveryAttempts* variable will be set to 0 due to the receipt of *TransactionEventRequest(triggerReason = "Deauthorized")*.

29 Situations where the "user interactions" on an EV lead to a non-emergency or error shutdowns may be included in the "Local" stopped reason. Similarly, "user interactions" that lead to emergency or erroneous shutdowns cannot be calculated as successful charge sessions including the cases where the user pushes the emergency shutdown button or the S3 switch has been activated since they would result in an Emergency and/or Error Shutdown per Table F3 in SAE J1772 and per Table 3.

CX006 – UnderVoltage	MREC
CX008 – EmergencyStop	MREC
CX010 – InvalidVehicleMode	MREC
CX014 – PilotFault	MREC
CX015 – PowerLoss	MREC
CX017 – EVSEContactorFault	MREC
CX019 – CableOverTempStop	MREC
CX020 – PartialInsertion	MREC
CX023 – ProximityFault	MREC
CX024 – ConnectorVoltageHigh	MREC
CX025 – BrokenLatch	MREC
CX026 – CutCable	MREC

3.4 Session Success

Session Success is defined as the percent of charge attempts[30] that successfully start a charging session (i.e., that result in an EVSE starting to deliver power to an EV) and the charging session goes on to successfully complete.

This KPI measures the fraction of all charge attempts made by all customers over a period of time that: (a) were successful; (b) the charging session completed without any errors that terminated or suspended charging unexpectedly; and (c) the customer was able to unplug without manual intervention to unlock the connector from the vehicle.

Session Success is measured as a percentage and applies to one or more charging ports at one or more charging stations.[31] There are two nominal use-cases: (1) Authorize after plug-in; and (2) Authorize before plug-in.

More details are presented below.

3.4.1 Authorize After Plug-In using OCPP 2.0.1

In this scenario, the EV/EV driver authorizes the transaction after being asked to plug-in and a session begins.

Per OCPP 2.0.1 Section C.2.1, there are eight ways in which a transaction may be authorized. Of these eight, only two are applicable in this scenario when using ISO 15118 and OCPP:

- C07 – Authorization using Contract Certificates
- C08 – Authorization at EVSE using ISO 15118 External Identification Means (EIM).

In both scenarios, a successful session for every plug-in attempt may be defined as the receipt of *TransactionEventRequest (transactionInfo:stoppedReason = "EnergyLimitReached" or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "StoppedByEV" or*

[30] *Charge attempts* are defined as when either: (a) an EV driver connects the EV to the EVSE; or (b) the EV driver begins authorization (provide valid credentials and/or payment) of a charging session.
[31] When parties implementing this KPI agree to exemptions, the Plug-In Attempts (or Attempts to Authorize) exempt may be subtracted from the denominator. In this case, each exemption should be tracked as a Plug-In Attempt (or Authorize Attempt) log paired with its eventual corresponding failure log indicating an agreed exempt failure reason.

“LocalOutofCredit” or “TimeLimitReached”) given it is followed by the receipt of *StatusNotificationRequest (connectorStatus ≠ ”Occupied”)*. Session Success may be defined as the ratio of number of *TransactionEventRequest (transactionInfo:stoppedReason = ”EnergyLimitReached” or “SOCLimitReached” or ”Local” or “Remote” or “EVDisconnected” or “StoppedByEV” or “LocalOutofCredit” or “TimeLimitReached”)* to *TransactionEventRequest(triggerReason = "CablePluggedIn", eventType = "Started")* message (Section 1.60.1 in OCPP 2.0.1):[32]

$$\text{Session Success} = \left(\frac{\begin{array}{c}\sum \text{TransactionEventRequest(transactionInfo: stoppedReason = "EnergyLimitReached"} \\ \text{or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "StoppedByEV" or} \\ \text{"LocalOutofCredit" or "TimeLimitReached")}\end{array}}{\sum \text{TransactionEventRequest(triggerReason = "CablePluggedIn", eventType = "Started")}} \right) \text{x100} \quad (12)$$

3.4.2 Authorize After Plug-In using OCPP 1.6J

In this scenario, the EV/EV driver authorizes the transaction after being asked to plug-in and a session begins.

Per OCPP 1.6J, there are two ways in which a transaction may be authorized:

- Operations Initiated by Charge Point (Section 4 in OCPP 1.6J)
- Operations Initiated by Central System (Section 5 in OCPP 1.6J).

In both scenarios, a successful session for every plug-in attempt may be defined as the receipt of *StopTransaction.req(Reason = ”EnergyLimitReached” or “SOCLimitReached” or ”Local” or “Remote” or “EVDisconnected” or “UnlockCommand”)** given it is followed by the receipt of *StatusNotification.req(status≠ ”Occupied”)*. Session Success may be defined as the ratio of number of *StopTransaction.req(Reason = ”EnergyLimitReached” or “SOCLimitReached” or ”Local” or “Remote”)** to *StatusNotification.req(status = "Preparing")*:[33]

[32] Situations where the “user interactions” on an EV lead to non-emergency or error shutdowns may be included in the “Local” stopped reason. Similarly, “user interactions” leading to emergency or erroneous shutdowns cannot be calculated as successful charge sessions including the cases where the user pushes the emergency shutdown button or the S3 switch has been activated since they would result in an Emergency and/or Error Shutdown per Table F3 in SAE J1772 and per Table 3.

[33] Situations where the “user interactions” on an EV lead to non-emergency or error shutdowns may be included in the “Local” stopped reason. Similarly, “user interactions” leading to emergency or erroneous shutdowns cannot be calculated as successful charge sessions including the cases where the user pushes the emergency shutdown button or the S3 switch has been activated since they would result in an Emergency and/or Error Shutdown per Table F3 in SAE J1772 and per Table 3.

$$\text{Session Success} = \left(\frac{\sum \text{StopTransaction.req(Reason = "EnergyLimitReached" or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "UnlockCommand")}}{\sum \text{StatusNotification.req(status = "Preparing")}} \right) \text{x } 100 \quad (13)$$

3.4.3 Authorize Before Plug-In using OCPP 2.0.1

In this scenario, the EV/EV driver authorizes the transaction before being asked to plug-in, the connector is unlocked, and the session begins. This is particularly challenging since the EV is not connected and no EV-EVSE communication is established. Per OCPP 2.0.1 Section C.2.1, there are eight ways in which a transaction may be authorized. Of these eight, only six are applicable in this scenario, as shown in Table 4. [34]

Table 4 outlines the message request-response pairs that may be tracked to calculate *Session Success* for each possible authorization method in each transaction given that they are followed by the receipt of *StatusNotificationRequest(connectorStatus≠ "Occupied")*.

Table 4. Relevant request-response pairs for tracking Session Success.

Scenario	Message Pair	Comments
C01 – EV Driver Authorization using RFID	TransactionEventRequest - AuthorizeResponse	Follow Eq. 14.
C02 – Authorization using a start button	-	The CSMS cannot reject the transaction. CSS = 100%.
C03 – Authorization using credit/debit card	Both	Follow Eq. 14 or Eq. 15 based on individual workflows.
C04 – Authorization using PIN-code	TransactionEventRequest - AuthorizeResponse	Follow Eq. 14.
C05 – Authorization for CSMS initiated transactions	TransactionEventRequest - RequestStartTransactionResponse	Follow Eq. 15.
C06 – Authorization using local id type	AuthorizeRequest - AuthorizeResponse	Follow Eq. 14.

[34] Currently, it is not possible to track all failed attempts that are directly made by the user using a payment terminal on the charger for which no OCPP message(s) are generated. Such failures may include authentication failures that may occur on the payment system due to several factors including Internet connectivity failures, timeouts, bad card readers, etc. Although these failures are indicated to the user on the payment terminal, the standard offers limited insights into such events and are deemed out-of-scope for the purpose of this document. It is recommended that the relevant standard bodies investigate these failures in more detail and address this gap in future editions of their standards.

Based on the request-response pairs given in Table 4, *Session Success* may be calculated as:[32]

$$\text{Session Success} = \left(\frac{\begin{gathered}\sum \text{TransactionEventRequest(transactionInfo: stoppedReason = "EnergyLimitReached"} \\ \text{or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "StoppedByEV"} \\ \text{or "LocalOutofCredit" or "TimeLimitREached")}\end{gathered}}{\begin{gathered}\sum \text{AuthorizeResponse(idTokenInfo: status = "Accepted")} + \\ \sum \text{AuthorizeResponse(idTokenInfo: status} \neq \text{"Accepted")}\end{gathered}} \right) \text{x 100} \quad (14)$$

Or

$$\text{Session Success} = \left(\frac{\begin{gathered}\sum \text{TransactionEventRequest(transactionInfo: stoppedReason = "EnergyLimitReached"} \\ \text{or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "StoppedByEV"} \\ \text{or "LocalOutofCredit" or "TimeLimitReached")}\end{gathered}}{\begin{gathered}\sum \text{RequestStartTransactionResponse(status = "Accepted")} + \\ \sum \text{RequestStartTransactionResponse(status = "Rejected")}\end{gathered}} \right) \text{x 100} \quad (15)$$

In some cases, it may be possible the transaction is "authorized" using the Local Authorization List (or any authorization cache). In such cases, it is possible the *AuthorizeRequest-AuthorizeResponse* pair in Eq. 14 is not encountered. Instead, a *TransactionEventRequest-Response* pair will be seen. In such cases, the use of Eq. 16 is recommended to calculate *Session Success*.

$$\text{Session Success} = \left(\frac{\begin{gathered}\sum \text{TransactionEventRequest(transactionInfo: stoppedReason = "EnergyLimitReached"} \\ \text{or "SOCLimitReached" or "Local" or "Remote" or "EVDisconnected" or "StoppedByEV"} \\ \text{or "LocalOutofCredit" or "TimeLimitReached")}\end{gathered}}{\begin{gathered}\sum \text{TransactionEventResponse(idTokenInfo: status = "Accepted")} + \\ \sum \text{TransactionEventResponse(idTokenInfo: status} \neq \text{"Accepted")}\end{gathered}} \right) \text{x 100} \quad (16)$$

3.4.4 Authorize Before Plug-In using OCPP 1.6J

In this scenario, the EV/EV driver authorizes the transaction before being asked to plug-in or the connector is unlocked. This is particularly challenging since the EV is not connected and no EV-EVSE communication is established. Per OCPP 1.6J, there are two ways in which a transaction may be authorized:

- Operations Initiated by Charge Point (Section 4 in OCPP 1.6J)
- Operations Initiated by Central System (Section 5 in OCPP 1.6J).

Table 5 outlines the message request-response pairs that may be tracked to calculate Charge Start Success for each possible authorization method in each transaction given that they are followed by the receipt of *StatusNotification.req(status≠"Occupied")*. [35]

Table 5. Relevant request-response pairs for tracking Session Success for OCPP 1.6J.

Scenario	Request-Response Pair	Comments
EV Driver Authorization using RFID	StatusNotification.req – Authorize.conf	Follow Eq. 17.
Authorization using a start button	-	The CSMS cannot reject the transaction. CSS = 100%.
Authorization using credit/debit card	Multiple	Follow Eq. 17 or Eq. 18 based on individual workflows.
Authorization using PIN-code	StatusNotification.req - Authorize.conf	Follow Eq. 17.
Authorization for CSMS initiated transactions	StatusNotification.req - RequestStartTransactionResponse	Follow Eq. 17.
Authorization using local authorization or cache	StatusNotification.req - Authorize.conf	Follow Eq. 18.

Based on the request-response pairs given in Table 5, *Session Success* may be calculated as:[32]

$$\text{Session Success} = \left(\frac{\sum \text{StopTransaction.req}(\text{Reason} = \text{"Local" or "Remote" or "EVDisconnected" or "UnlockCommand"})}{\sum \text{Authorize.conf}(\text{idTagInfo: status} = \text{"Accepted"}) + \sum \text{Authorize.conf}(\text{idTagInfo: status} \neq \text{"Accepted"})}\right) \text{x } 100 \tag{17}$$

Or

$$\text{Session Success} = \left(\frac{\sum \text{StopTransaction.req}(\text{Reason} = \text{"Local" or "Remote"})}{\sum \text{RemoteStartTransaction.conf}(\text{status} = \text{"Accepted"}) + \sum \text{RemoteStartTransaction.conf}(\text{status} = \text{"Rejected"})}\right) \text{x } 100 \tag{18}$$

In some cases, it may be possible the transaction is "authorized" using the Local Authorization List (or any authorization cache). In such cases, it is possible the *Authorize.req- conf* pair in Eq. 17 is not encountered. Instead, a *StartTransaction.req-conf* pair will be seen. In such cases, the use of Eq. 19 is recommended to calculate *Session Success*.

[35] Currently, it is not possible to track all failed attempts that are directly made by the user using a payment terminal on the charger for which no OCPP message(s) are generated. Such failures may include authentication failures that may occur on the payment system due to several factors including Internet connectivity failures, timeouts, bad card readers, etc. Although these failures are indicated to the user on the payment terminal, the standard offers limited insights into such events and are deemed out-of-scope for the purpose of this document. It is recommended that the relevant standard bodies investigate these failures in more detail and address this gap in future editions of their standards.

$$\text{Session Success} = \left(\frac{\sum \text{StopTransaction.req}(\text{Reason} = \text{"Local" or "Remote" or "EVDisconnected" or "UnlockCommand"})}{\sum \text{StartTransaction.conf}(\text{idTagInfo: status} = \text{"Accepted"}) + \sum \text{StartTransaction.conf}(\text{idTagInfo: status} \neq \text{"Accepted"})} \right) \text{x } 100 \qquad (19)$$

4. Next Steps

This document details one way to implement the Interim KPIs leveraging OCPP messages. The following recommendations are opportunities where government and industry can work together to further the development and establishment of these KPIs:

- Implement interim set of KPIs into the EVerest Project[36]
- Identify the necessary data to calculate the ideal set of KPI and develop detailed instructions on how to implement the ideal set of KPIs

  Work with industry partners to implement the ideal set of KPIs
- Work with a standards development organization to codify the KPIs in a formal standard

[36] https://lfenergy.org/projects/everest/

CHARGEX
consortium

About the ChargeX Consortium

The National Charging Experience Consortium (ChargeX Consortium) is a collaborative effort between Argonne National Laboratory, Idaho National Laboratory, National Renewable Energy Laboratory, electric vehicle charging industry experts, consumer advocates, and other stakeholders. Funded by the Joint Office of Energy and Transportation, the ChargeX Consortium's mission is to work together to measure and significantly improve public charging reliability and usability by June 2025. For more information, visit **chargex.inl.gov**.

INL/RPT-24-77389

Argonne NATIONAL LABORATORY

INL Idaho National Laboratory

NREL Transforming ENERGY