Open Charge Metering Format

ENTWURF

Revision: 1.0

Mitwirkende

Rolle	Verantwortlicher	Firma
Verfasser	Andreas Mull, ABL (AM), Daniel Müller (DM)	ABL
Mitwirkung	Gerhard Weidinger	KEBA
Mitwirkung	Martin Klässner, Roland Angerer	has.to.be
Mitwirkung	Andreas Weber	Allego
Mitwirkung	Michael Staubermann	Webolution

Revisionsübersicht

Änderungen gegenüber der vorherigen Version sind Änderungsnachverfolgung gekennzeichnet.

Revision	Inhalt	Datum
1.0	Finaler Stand zur Integration in Transparenzsoftware. Feedback aus SAFE AG-Technik eingearbeitet: Signatur-Algorithmus doch nicht in Data-to-be-Signed-Bereich, damit theoretisch mehrere unterschiedliche Signaturen unabhängig voneinander angehängt werden können. Reading-Type optional. Zusätzlicher Zustand beim Status der Zeit: Relative Zeitabrechnung basierend auf Info-Uhr. OBIS-Codes erweitert auf anwendbare Möglichkeiten entsprechend Kommentaren von Herrn Weber.	21.02.2019/AM
0.5	Weiteres Feedback aus DKE/GAK 461.0.21 AP5 von Januar 2019 eingearbeitet: Klarstellungen zur Paginierung, Bezug von Seriennummern, Ladepunkt und Public-Key. Optionale Nutzdatensektion zur Zuordnung des Ladepunkts. Fehlerindex ersetzt durch fehlerhafte Größen; Fehler während Ladung als Ablesungsgrund. Feedback von Herrn Weber zu Version 0.3 eingearbeitet: Einspeißung, Stromarten. Byte-Offset bei den Binärbeschreibungen entfernt, da sich diese mit den StrEnum-Typen widersprachen. Feedback von Herrn Staubermann zur Elliptic-Curve-Kryptografie: Synonyme Namen der Kurven, Korrektur der Blocklängen bei den Public-Key-Typen.	11.02.2019/AM
0.4	Ergebnisse und Klarstellungen nach Feedback aus Telefonkonferenzen mit DKE/GAK 461.0.21 AP5: Dokument- und Format-Versionen korrigiert; Binärformat klargestellt: Network-Byte-Order; Zählung der Paginierung klargestellt, Umbruch definiert; Identifikation der Signatureinrichtung (Gateway) über Zähleridentifikation gestellt, in allgemeine Angaben integriert, Vendor zu Gateway umbenannt, Seriennummer entweder für Gateway oder Zähler obligatorisch; Benutzerzuordnung: Status aufgeteilt in generell ja/nein und Zuordungs-Level, Level in Gruppen eingeteilt; alle weiteren Angaben zum Status außer ja/nein sind nun optional. Verwendung Ereignisszähler klargestellt, Umbruch definiert; Hinweis auf Genauigkeit bei Messwert; Signatur-Algorithmus in den signierten Bereich verschoben; Signaturalgorithmen um zukünftige Kandidaten erweitert, Schlüsseltypen analog; Defaults mit OCMF-Version versehen. Autorisierungsmerkmale entsprechend OCPP 2.0 erweitert; Erweiterungspunkte für OCMF in Nutzdaten- und Signatursektion festgelegt.	11.01.2019/AM
0.3	Zustand des Zählers hinzugefügt für falsch ausgelesene Werte	11.12.2018/DM
0.2	Binärformat entsprechend der Umstellungen mit 0.1 angepasst, Beschreibung des Signatur-Algorithmus konkretisiert, StrEnum-Typ als flexiblen Erweiterungspunkt ins Binärformat eingeführt.	16.08.2018/AM

0.1	Dieses Dokument basiert auf ABL-Datenformat Version 1.14 und löst dieses ab. Initiale Version nach Diskussion des ABL-Datenformats (v1.13) mit KEBA und has.to.be: Entkopplung des binären Übertragungsformats aus dem eichrechtlich relevanten Teil, JSON als primäres Übertragungsformat. Header und Gesamtstruktur definiert. Felder abgeglichen mit dem Entwurfsstand „Anforderungen an die Datenstruktur für Messwerttupel in der Elektromobilität“ vom 26.04.2018. JSON-Format strukturell vereinfacht, Keys verkürzt. Möglichkeit mehrerer Ablesungen in einem Datensatz geschaffen. Firmware-Version des Zählers optional hinzugefügt. Fehler als Abbruch-Gründe für Ladevorgänge. Hash-Typ für Signatur definiert.	09.08.2018/AM

1 Allgemeines

1.1 Ziel

Ziel dieses Konzepts ist es, ein unabhängiges und allgemein verwendbares Datenformat zur Erfassung eichrechtlich relevanter Zählerablesungen von Ladestationen zu beschreiben. Ferner kann dieses Konzept als Diskussionsgrundlage für eine Verwertung im DKE dienen. Außerdem soll es zur Implementierung der Auswertung und Signaturprüfung des Formats durch die im Rahmen der SAFE-Initiative zu schaffende Transparenz-Software dienen.

Zum Zeitpunkt der Erstellung dieses Dokuments ist kein einheitliches Datenformat zur Übermittlung von signierten Zählerständen auf Normungsebene verabschiedet. Eine entsprechende Anwendungsrichtlinie ist in Entstehung befindlich. Das vorgelegte Datenformat entstammt einem internen Entwurf von ABL und wurde mit has.to.be und KEBA diskutiert.

Ein einheitliches Format bringt den Vorteil der Interoperabilität und ermöglicht eine einheitliche Verifizierungs-Software beim Betreiber und dem Endbenutzer.

Das hier dargestellte Konzept orientiert sich an dem *Ende-zu-Ende Sicherungskonzept für eine eichrechtlich günstige Lösung („GL“) für die Übertragung von Messdaten auf Fernanzeigen* von Dr. Martin Kahmann [MEMO-GL]. Weiter baut es auf das ABL-interne Konzept zur Eichrechtskonformität auf.

Dieses Dokument ist wie folgt gegliedert: Zunächst wird eine Einordnung und Darstellung der Übermittlung des Datenformats gegeben, in den weiteren Sektionen wird das Datenformat erläutert.

1.2 Lizenz

TODOs:

- Lizenz definieren
- MIT: sehr frei, Verwertung in Werk enthaltener Patente nicht geregelt
- Apache: auch sehr frei, soll jedoch Regelung zu Patenten enthalten -> Klären!

1.3 Referenzen

OCPP 1.5: Open Charge Alliance: Open Charge Point Protocol – Interface description between Charge Point and Central System, Version 1.5, Stand: 8. Juni 2012

MEMO-GL: Messsysteme für Ladeeinrichtungen: Ende-zu-Ende-Sicherungskonzept für eine eichrechtlich günstige Lösung („GL“) für die Übertragung von Messdaten auf Fernanzeigen, Dr. Martin Kahmann, Braunschweig

OBIS: IEC 62056-6-1 und -6-2.

JSON: Einführung in JSON, https://www.json.org/json-de.html (Abruf: 2018-04-04)

1.4 Verwendete Abkürzungen

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CDR	Charge Data Record
CPO	Charge Point Operator, Betreiber der Ladestation
DKE	Deutsche Kommission Elektrotechnik Elektronik Informationstechnik
ECDSA	Elliptic Curve Digital Signature Algorithm
EMP	Electric Mobility Provider, Fahrstromanbieter
ISO	International Standards Organisation
MID	Measuring Instruments Directive, europäische Richtlinie 2004/22/EG
OBIS	OBject Identification System
OCPP	Open Charge Point Protocol
PIN	Personal Identification Number
PTB	Physikalisch-Technische Bundesanstalt
RFC	Request For Comments
RFID	Radio Frequency Identification
RTU	Remote Terminal Unit
SHA	Secure Hash Algorithm
TLS	Transport Layer Security
UID	Unique IDentification

UTC Universal Time Coordinated

2 Übermittlung signierter Zählerstände

2.1 Einbettung in OCPP

Bereits OCPP Version 1.5 erlaubt im Rahmen der MeterValue.req-Nachricht die simultane Übermittlung einer beliebigen Anzahl von MeterValue-Objekten. In jedem solchen Objekt kann wiederum ein Zeitstempel zusammen mit einem oder mehreren Ablesungswerten notiert sein. Jeder Wert kann optional durch Attribute begleitet werden, eines davon vom Enum-Typ ValueFormat, der zwei mögliche Werte besitzt:

- Raw: Data is to be interpreted as integer/decimal numeric data.
- SignedData: Data is represented as a signed binary data block, encoded as hex data.

Quelle: [OCPP 1.5]

Damit ist es möglich, die von OCPP vorgesehene Funktionalität weiterhin zu nutzen und mit der Möglichkeit der Übermittlung von signierten Zählerwerten zu kombinieren.

Das hier vorgestellte Datenformat ist folglich gedacht zur parallelen Übermittlung eines solchen SignedData-„Wertes“ neben den bisherigen Raw-Werten.

Während die Ablesung der bisherigen Werte evtl. mit einer nicht eichrechtskonformen (OCPP-basierten) Zeit geschieht, basieren die signierten Zählerwerte jedoch auf einer solchen. Sogar die zwei resultierenden Zeiten können getrennt in zwei MeterValue-Objekten korrekt dargestellt werden.

Dieser Mechanismus wird genutzt im Rahmen von:

- Beginn einer Transaktion (StartTransaction)
- Ablesungen während einer Transaktion (MeterValues)
- Ende der Transaktion (StopTransaction)
- Charge Data Record über die gesamte Transaktion mit Werten zu Begin und Ende (optional)
- Ablesungen für sogenanntes „Fiscal-Metering“ (MeterValuesAlignedData): Dabei wird kein Bezug zu laufenden Transaktionen erfasst, sondern nur die absoluten Zählerwerte aller Ladepunkte zu vorher bestimmten Zeitpunkten.

Anzumerken ist, dass in der Regel eichrechtlich relevant folgende Ablesungen sind:

- Begin- und Endwert im Bezug auf den Ladevorgang
- Tarifwechsel (gängige Praxis nur zur Viertelstunde der Uhr)

Damit ist folgende OCPP-Konfiguration typischerweise sinnvoll:

- ClockAlignedDataInterval=900 (15min)
- MeterValuesAlignedData=Active.Energy.Register.Import
- Eine Controller-Software sorgt für die Erzeugung zusätzlicher MeterValues mit signierten Werten zu Beginn und Ende des Ladevorgangs über den simplen Integer-Wert hinaus, der Bestandteil der StartTransaction/StopTransaction-Nachrichten ist.

2.2 Signierungs- und Verifikations-Prozess

Um einen Datensatz zu prüfen, müssen zunächst die Daten, welche signiert wurden, wieder in ihrer Ursprungsform hergestellt werden. Auf dieser Ursprungsform erfolgt eine Hash-Berechnung. Das Resultat wird über die digitale Signatur per ECDSA geprüft.

Zur Herstellung der Ursprungsform werden Header und Signatur werden dazu vom Datensatz abgetrennt. Liegt der Datensatz im Darstellungsformat vor (mit White-Space) muss dieser ins JSON-basierte Übertragungsformat umgewandelt werden (Entfernung von White-Space), da dieses Format das Ursprungsformat bildet. Liegt der Datensatz im binären Übertragungsformat vor, muss er zuerst ins JSON-basierte Übertragungsformat umgewandelt werden. Ein im Binärformat eventuell vorliegender Public-Key wird dazu ebenfalls abgetrennt. Bei JSON-basierter Übertragung kommt der Public-Key über einen getrennten Übertragungsweg zur Prüfung.

Damit sind im der vereinheitlichte Datensatz, die Signatur und der Public-Key voneinander getrennt dargestellt.

Der Zusammenhang zwischen mehreren einzelnen Datensätzen wird durch die durchgehende Paginierung gewährleistet. Diese ist neben der Signatur von einer Prüfkomponente zu verifizieren. Der erste Datensatz muss als Beginn eines Ladevorgangs, der letzte als Ende des Ladevorgangs gekennzeichnet sein. Dazwischen dürfen keine Datensätze entnommen oder hinzugefügt worden sein. Ebenso müssen zwischenzeitliche Fehlerzustände (Fehlerzähler) und das Auffinden von nicht verwendbaren Größen zu einem Fehler bei der Prüfung führen. Außerdem müssen alle Datensätze von derselben Quelle (Seriennummer) stammen.

3 Datenformat

Das Datenformat soll möglichst einfach aufgebaut und vom versierten Benutzer nachvollziehbar sein. Aus Herstellersicht ist ein modulares Datenformat wünschenswert, damit zukünftige Erweiterungen umgesetzt werden können. Allerdings muss im Rahmen der Übertragung der signierten Datensätze auch das zur Verfügung stehende Datenvolumen betrachtet werden, was in einer weiteren Anforderung resultiert, der Kompaktheit des Datensatzes. Ein weiteres Anliegen ist, den Prüfprozess möglichst generisch, d.h. auch für andere Quellformate anwendbar, zu gestalten.

Um all diese Anforderungen zu erfüllen, werden anstelle eines Datenformats mehrere Teilformate definiert:

1. Ein Darstellungsformat, JSON-basiert (nachvollziehbar)
2. Ein Übertragungsformat, JSON-basiert (vereinheitlicht)
3. Ein kompakteres Übertragungsformat zur optionalen Verwendung als strukturiertes Binärformat (kompakt)

Alle Formate bestehen aus mehreren Sektionen:

1. Header zur eindeutigen Kennzeichnung des Formats
2. Abschnitt mit den eigentlichen Nutzdaten
3. Signatur über die Nutzdaten
4. Optional: Public-Key

Das JSON-Übertragungsformat dient als Eingabeformat in die Transparenz-Software.

Falls bei der Kommunikation zwischen Ladestation und Backend Datenvolumen eingespart werden soll ist die Binärformat-Variante des Übertragungsformats hilfreich. Es ist denkbar, dass das Backend dieses Format per OCPP-ChangeConfiguration-Methode auf der Ladestation auswählt. Wird es benutzt, muss das Backend Sorge dafür tragen, dass für die Transparenz-Prüfung wieder das JSON-Übertragungsformat aus dem Binärformat erzeugt wird. Dazu wird es einen Parser geben.

In den folgenden Kapiteln wird das Datenformat für die Darstellung und Übertragung der signierten Zählerwerte definiert.

4 Darstellungsformat und JSON-basiertes Übertragungsformat

Für das Darstellungsformat wurde JSON-Notation zur besseren Lesbarkeit und einfacheren Nachvollziehbarkeit gewählt. Um bereits hier Datenvolumen einzusparen, wurden folgenden Maßnahmen getroffen:

- Bezeichner sind kurz gehalten (da einige ohnehin erklärungsbedürftig sind, geht hier nicht viel Klarheit verloren)
- An einigen Stellen wurde bewusst auf Kapselung verzichtet. Anstelle sind die Bezeichner kanonisiert.

Dass das Darstellungsformat entsprechend der JSON-Notation mit Leerzeichen und Zeilenvorschüben an beliebiger Stelle, genannt White-Space, zur Darstellung angereichert werden kann erleichtert es einem Nutzer zudem die Lesbarkeit.

Das JSON-basierte Übertragungsformat hat zusätzlich folgende Eigenschaft:

- Eindeutiger Kennzeichner ist vorangestellt.
- White-Space wird eliminiert.
- Signatur wird angehängt.

Für die Übertragung und die Signaturoperationen werden die Nutzdaten im Darstellungsformat durch einen Vereinheitlicher gekürzt, der sämtliches optionales White-Space aus dem JSON entfernt. Nach Bildung der Signatur wird diese in einer separaten Sektion angehängt. Optional kann ebenso der zugehörigen Public-Key mit übertragen werden. Vorangestellt wird ein Header, der das Übertragungsformat eindeutig kennzeichnet. Damit liegt das JSON-basierte Übertragungsformat vor.

Die Signatur wird lediglich über die Nutzdatensektion im vereinheitlichten Darstellungsformat gebildet. D.h. evtl. zur Seite gestellte Information über die Beschaffenheit des Darstellungsformats (Header) geht nicht in die Signaturprüfung mit ein.

Im Folgenden werden die verschiedenen Abschnitte des Darstellungsformats (und damit des JSON-basierten Übertragungsformats) erläutert. Weiterhin wird ein Beispiel gegeben in lesbarer und verkürzter Form.

4.1 Sektionen

Am Anfang des Übertragungsformats steht ein Header zur eindeutigen Kennzeichnung des Formats.

Durch Pipe-Zeichen getrennt folgen Nutzdatensektion und Signatursektion.

Damit ergibt sich für die Sektionierung folgender Aufbau:

OCMF|Nutzdatensektion|Signatursektion

Dabei sind die Worte Nutzdatensektion und Signatursektion durch die Inhalte der jeweiligen Sektionen zu ersetzen.

4.2 Bezug von Seriennummern, Ladepunkt und Public-Key

Zum Aufbau eines Bezugs zwischen dem zur Signaturprüfung zu benutzenden Public-Key und des Ladepunktes dienen eine oder mehrere Seriennummern, alternativ kann auch eine direkte Indentifikation des Ladepunktes per ID erfolgen.

Die Public-Keys zum Ladepunkt müssen auf anderem Wege als diesem Protokoll (Out-of-Band) an die Prüfkomponente in Verbindung mit der verwendeten ID oder den Seriennummern übermittelt werden, z.B. über ein zentrales Register.

Um eine Signaturkomponente eindeutig zu identifizieren werden in diesem Protokoll folgende Merkmakle benutzt:

- Ist die Signaturkomponente dem Energiezähler zuzurechnen, reicht dessen Seriennummer.
- Ist die Signaturkomponente selbst das Gateway und nur ein Ladepunkt wird bedient, reicht die Seriennummer des Gateways oder des Energiezählers.
- Ist ein Gateway für mehr als einen Ladepunkt zuständig, wird die Kombination aus den Seriennummern von Gateway und Energiezähler zur eindeutigen Identifikation herangezogen.
- Alternativ kann eine direkte Identifikation des Ladepunktes erfolgen.

Da diese Bedingungen vom Aufbau der Ladestation abhängen, sind die entsprechenden Seriennummernfelder weiter unten als bedingt obligatorisch gekennzeichnet.

4.3 Nutzdatensektion

Die Nutzdatensektion besteht aus einem JSON-Objekt. Darin enthalten sind mehrere Wertzuordnungen bzw. Unter-Objekte. Die Reihenfolge der Objekte innerhalb des Nutzdatenabschnitts darf zwischen Signierung und Verfikation der Signatur nicht verändert werden.

Die Felder müssen stets in der hier dargestellten Reihenfolge erzeugt werden, damit eine für die Signaturprüfung Erfolg versprechende Rückführung aus dem Binärformat möglich ist.

4.3.1 Allgemeine Angaben

Die Angaben in diesem Abschnitt beziehen sich vorzugsweise auf die signaturerzeugende Komponente, welche hier der Einfachheit halber als Gateway bezeichnet wird. Je nach Aufbau ist evtl. die Bezeichnung Messkapsel oder Signierkomponente treffender.

Key:	Typ	Beschreibung:
FV	String	Format-Version: Version des Datenformats in der Darstellung <major>.<minor> Die Versionsangabe wird entsprechend der Version dieses Dokumentes codiert, d.h. 0.4 entspricht Major 0 und Minor 4.
GI	String	Gateway-Identification: Bezeichner des Herstellers für das System, welches die vorliegenden Daten erzeugt hat (Hersteller, Modell, Variante, etc.).
GS	String	Gateway-Serial: Seriennummer des o.g. Systems Dieses Feld ist bedingt obligatorisch.
GV	String	Gateway-Version: Versionsbezeichnung des Herstellers für die SW auf o.g. System Dieses Feld ist optional.

Tabelle 1: Felder zu allgemeinen Angaben

4.3.2 Paginierung

Die Paginierung gibt den Bezug zu einer Transaktion an und ordnet den Gesamtdatensatz im Strom der Ablesungen über die Zähler nachvollziehbar ein. Für die Paginierung stehen mehrere Kontexte zur Verfügung, jeder Kontext hat einen eigenen Zählerstand. Der jeweilige Paginierungszähler wird monoton aufsteigend mit einem Inkrement von 1 gezählt. D.h. aufeinander folgende Werte des Paginierungszählers zählen fortlaufend im Raum der natürlichen Zahlen.

Bedingt durch die Kompatibilität mit dem Binärformat (siehe unten) muss der Paginierungszähler nach dem Wert 2^{32}-1 auf 0 zurückgesetzt werden. Zusammen mit dem Zeitpunkt der Ablesung in einem Datensatz ist dieser Umbruch jedoch nachvollziehbar. Außerdem findet er ausreichend selten statt: Für einen erzeugten Datensatz pro Sekunde dauert es ca. 136 Jahre bis zum ersten Umbruch.Der Kontext für Transaktionen ist in jedem Fall zu unterstützen. Der Kontext Fiscal ist optional und kann von der Signaturkomponente optional unterstützt werden, falls auch außerhalb von Transaktionen signierte Ablesungen gewünscht sind.

Key:	Typ:	Beschreibung:
PG	String	Paginierung des gesamten Datensatzes, d.h. der Daten, die gemeinsam unter eine Signatur fallen. Format: <Kennzeichen><Zahl> Der String setzt sich aus einem kennzeichnenden Buchstaben für den Kontext und einer Zahl ohne führende Nullen zusammen. Für jeden Kontext existiert ein eigener unabhängiger Paginierungszähler. Folgende Kennzeichen sind definiert: F: Fiscal – Ablesungen unabhängig von Transaktionen (optional) T: Transaction – Ablesungen im Transaktionsbezug (obligatorisch) Der jeweilige Paginierungszähler wird nach jeder Verwendung für einen Datensatz erhöht.

Tabelle 2: Paginierungsfeld

4.3.3 Zähleridentifikation

Die Zähleridentifikation ist optional, wenn die signaturerzeugende Komponente bereits in den allgemeinen Daten identifiziert wurde.

Key:	Typ:	Beschreibung:
MV	String	Meter-Vendor: Hersteller-Identifikation des Zählers, Name des Herstellers Dieses Feld ist optional.
MM	String	Meter-Model: Modell-Identifikation des Zählers Dieses Feld ist optional.
MS	String	Meter-Serial: Seriennummer des Zählers Dieses Feld ist bedingt obligatorisch.

MF	String	Meter-Firmware: Firmware-Version des Zählers Dieses Feld ist optional.

Tabelle 3: Felder zur Identifikation des Zählers

4.3.4 Benutzerzuordnung

Hierzu gehörige Felder sind genau dann enthalten, wenn Transaktionsbezug besteht. Auch wenn bei Transaktionsbezug noch kein Nutzer zugeordnet werden kann, ist der Abschnitt enthalten. Wenn kein Transaktionsbezug besteht, fehlt diese Gruppe von Feldern.

Key:	Typ:	Beschreibung:
IS	Boolean	Identification-Status: Genereller Status zur Benutzerzuordnung: true: Benutzer erfolgreich zugeordnet, false: Benutzer nicht zugeordnet.
IL	String	Identification-Level: Aufgeschlüsselter Gesamtstatus der Benutzerzuordnung, dargestellt durch einen Bezeichner aus Tabelle 23 Dieses Feld ist optional.
IF	Array of String	Identification-Flags: Detailaussagen zur Nutzerzuordnung, dargestellt durch einen oder mehrere Bezeichner aus Tabelle 25 bis Tabelle 28. Die Bezeichner werden stets als String-Elemente in einem Array notiert. Auch ein oder kein Element muss als Array notiert werden. Dieses Feld ist optional.
IT	String	Identification-Type: Typ der Identifikationsdaten, Bezeichner siehe Tabelle 29
ID	String	Identification-Data: Die eigentlichen Identifikationsdaten entsprechend des Typs aus Tabelle 29, also z.B. eine Hex-Codierte UID entsprechend ISO14443.

Tabelle 4: Felder für die Benutzerzuordnung

4.3.5 Zuordnung des Ladepunktes

Diese optionale Nutzdatensektion bietet eine Möglichkeit zur Identifikation des Ladepunktes. Dies kann eine Alternative zur Identifikation durch Seriennummern darstellen oder zusätzlich genutzt werden.

Key:	Typ:	Beschreibung:
CT	String	Charge-Point-Identification-Type: Typ der Angabe zur Identifikation des Ladepunktes, Bezeichner siehe Tabelle 30
CI	String	Charge-Point-Identification: Identifikations-Angabe für den Ladepunkt.

Tabelle 5: Felder für die Identifikation des Ladepunktes

4.3.6 Ablesungen

Eine oder mehrere Ablesungen können in einem Datensatz abgelegt werden. Jede Ablesung wird in einem Sub-Objekt gekapselt. Diese Objekte werden als Array unter dem Key „RD“ für Reading(s) notiert. Dieses Array enthält somit ein oder mehrere anonyme Objekte.

Jedes dieser Objekte ist aufgebaut, wie in Tabelle 6 beschrieben.

Bei den Ablesungen gilt, dass Felder, die einen identischen Wert zur vorherigen Ablesung haben, ausgelassen werden. Dies gilt allerdings nur innerhalb eines signierten Datensatzes. Das bedeutet konkret, dass beispielsweise nur im ersten Sub-Objekt das Feld „RI“ mit einem Wert (hier dem OBIS-Code) definiert wird und damit für alle weiteren Ablesungen gültig ist. Ebenso kann zum Beispiel das Feld „TX“ im ersten Sub-Objekt als „B“ definiert werden, im zweiten als „C“, im letzten mit „E“, was bedeutet dass die Ablesungen drei bis zur vorletzten den Wert „C“ übernehmen.

Die Felder „RV“, „RI“ , „RU“ und „RT“ können ausgelassen werden, falls nur auf den Eintritt eines Fehlerzustands (Ereignis) des Zählers hingewiesen werden soll.

Die Felder „RI“ und „RU“, bilden eine Gruppe. Felder einer Gruppe sind entweder alle gemeinsam präsent oder werden gemeinsam ausgelassen.

Ein Stromausfall während zeitbasierter Abrechnung stellt ein Fehlerereignis dar.

Key:	Typ:	Beschreibung:
TM	String	Time: Angabe zur Systemzeit der Ablesung und Synchronisationszustand. Die Zeit wird nach ISO 8601 mit einer Auflösung von Millisekunden beschrieben. Das Format wird dementsprechend nach folgendem Schema gebildet: <Jahr >-<Monat>-<Tag>T<Stunden>:<Minuten>:<Sekunden>,<Millisekunden><Zeitzone> Dabei wird das Jahr vierstellig dargestellt, Monat, Tag, Stunden, Minuten und Sekunden zweistellig, Millisekunden dreistellig. Die Angabe zur Zeitzone besteht aus einem Vorzeichen und einer vierstelligen Angabe für Stunden und Minuten. Beispiel: 2018-07-24T13:22:04,000+0200 Der Synchronisationszustand besteht aus einem Großbuchstaben als Bezeichner. Dieser wird, getrennt durch ein Leerzeichen, der Zeit hinangestellt. Verfügbare Zustände siehe Tabelle 31.
TX	String	Transaction: Ablesungsgrund, Bezug der Ablesung zur Transaktion, notiert als Großbuchstabe: B – Beginn der Transaktion C – Charging = während der Ladung (kann optional verwendet werden) X – Exception = Fehler während der Ladung, Transaktion wird fortgesetzt, Zeit und/oder Energie sind ab dieser Ablesung (einschl.) nicht mehr verwertbar. E – Ende der Transaktion, alternativ genauere Codes: L – Ladevorgang wurde lokal beendet R – Ladevorgang wurde aus der Ferne beendet A – (Abort) Ladevorgang wurde durch Fehler abgebrochen P – (Power) Ladevorgang wurde durch Stromausfall beendet S – Suspended = Transaktion aktiv, aber gerade keine Ladung (kann optional verwendet werden) T – Tarifwechsel Dieses Feld fehlt, wenn kein Transaktionsbezug vorhanden ist. (Fiscal Metering)
RV	Number	Reading-Value: Der Wert der Ablesung Hier wird auf das JSON-Datenformat Number zurückgegriffen, dies erlaubt unter anderem eine genaue Auszeichnung der gültigen Nachkommastellen. Die Darstellung darf durch weitere Behandlungsmethoden (z.B. Verarbeitung durch JSON-Parser) jedoch nicht umgeformt werden (Umschreibung der Zahl mit anderem Exponenten, Kürzung von Nachkommastellen, etc.) da damit die Darstellung der physikalischen Größe und damit potentiell die Anzahl der gültigen Stellen verändert würde. Entsprechend der Anwendungsregel wird empfohlen, den Messwert mit zwei Nachkommastellen Genauigkeit darzustellen, falls es sich um kWh handelt.
RI	String	Reading-Identification: Bezeichner, welche Größe abgelesen wurde, gemäß OBIS-Code.
RU	String	Reading-Unit: Einheit der Ablesung, z.B. kWh.
RT	String	**Reading-Current-Type: Die vom Zähler gemessene Stromart, z.B. Wechselstrom oder Gleichstrom, gemäß Tabelle 34: Vordefinierte Stromarten** . Dieses Feld ist optional. Es ist kein Default-Wert definiert.
EF	String	Error-Flags: Aussage, welche Größen durch einen Fehler nicht mehr verwertbar zur Abrechnung sind. Jedes Zeichen in diesem String kennzeichnet eine Größe. Folgende Zeichen sind definiert: E – Energie t – Zeit
ST	String	Status: Zustand des Zählers zum Zeitpunkt der Ablesung. Notiert als Kürzel gemäß Tabelle 22.

Tabelle 6: Felder eines Ablesungs-Objekts

4.3.7 Erweiterungspunkte

In der Nutzdatensektion ist es für verschiedene Hersteller eventuell nötig, dass diese eigene Daten hinzufügen können. Dazu werden im Namensraum des JSON-Objekts folgende Erweiterungspunkte zur freien Benutzung in Abhängigkeit vom Hersteller reserviert:

- Reserviert sind alle Namen auf oberster Ebene im JSON-Objekt der Nutzdatensektion, welche mit folgenden Buchstaben beginnen:
  - U
  - V
  - W
  - X
  - Y
  - Z
- Da in der Nutzdatensektion nachträglich keine Veränderungen oder Hinzufügungen erfolgen können (sic), sind keine entsprechenden Erweiterungspunkte definiert. Siehe unten, Sektion 4.4.2.

4.4 Signatursektion

Die Signatursektion besteht ebenfalls aus einem eigenständigen JSON-Objekt. Darin enthalten sind mehrere Wertzuordnungen bezüglich der Signaturdaten.

4.4.1 Signaturdaten

Key:	Typ:	Beschreibung:
SA	String	Signature-Algorithm: Wählt den zur Signaturbildung verwendeten Algorithmus aus. Dies umfasst den Signatur-Algorithmus, dessen Parameter und den Hash-Algorithmus der auf den zu signierenden Daten angewendet wird. Diese Angabe ist optional. Wird sie ausgelassen, ist der Default-Wert gültig. Tabelle 35 zeigt, welche Werte möglich sind.
SE	String	Signature-Encoding: Bezeichnet, wie die Signaturdaten kodiert sind, um im JSON-String abgelegt werden zu können. Diese Angabe ist optional. Wird sie ausgelassen, ist der Default-Wert gültig. Folgende Werte sind möglich: hex – Die Signaturdaten sind im JSON-String hexadezimal codiert dargestellt (Default) base64 – Die Signaturdaten sind entsprechend base64 abgelegt
SM	String	Signature-Mime-Type: Bezeichnet, wie die Signaturdaten zu interpretieren sind. Diese Angabe ist optional. Wird sie ausgelassen, ist der Default-Wert gültig. Folgende Werte sind möglich: application/x-der – Ablage als ASN.1-codierte .der-Datei (Default)
SD	String	Signature-Data: Die eigentlichen Signaturdaten entsprechend obiger Formatangaben.

Tabelle 7: Signaturdatenfelder der Signatursektion

4.4.2 Erweiterungspunkte

Analog zu den Erweiterungspunkten in Sektion 0, sind eventuell auch ungeschützte Erweiterungspunkte wünschenswert. Folgende sind dazu reserviert:

- Für die Verwendung durch Hersteller sind alle Namen auf oberster Ebene im JSON-Objekt der Signaturdatensektion reserviert, welche mit folgenden Buchstaben beginnen:
  - U
  - V
  - W
  - X
  - Y
  - Z
- Für die Verwendung durch Komponenten in der nachgelagerten Verarbeitung von Daten sind entsprechend folgende Anfangsbuchstaben reserviert:
  - A

- B
- C
- D
- E
- F

4.5 Beispiel

Beispiele zu den resultierenden JSON-basierten Formaten:

```
OCMF|{
    "FV": "1.0",
    "GI": "ABL SBC-301",
    "GS": "808829900001",
    "GV": "1.4p3",

    "PG": "T12345",

    "MV": "Phoenix Contact",
    "MM": "EEM-350-D-MCB",
    "MS": "BQ27400330016",
    "MF": "1.0",

    "IS": true,
    "IL": "VERIFIED",
    "IF": [ "RFID_PLAIN", "OCPP_RS_TLS" ],
    "IT": "ISO14443",
    "ID": "1F2D3A4F5506C7",

    "RD": [
        {
            "TM": "2018-07-24T13:22:04,000+0200 S",
            "TX": "B",
            "RV": 2935.6,
            "RI": "1-b:1.8.0",
            "RU": "kWh",
            "RT": "AC",
            "EF": "",
            "ST": "G"
        }
    ]
}|{
     "SD": "887FABF407AC82782EEFFF2220C2F856AEB0BC22364BBCC6B55761911ED651D1A922BADA88818C9
671AFEE7094D7F536"
}
```

Abbildung 1: Beispieldatei Darstellungsformat

Beispiel zum vereinheitlichten Darstellungsformat:

```
TODO
x
```

Abbildung 2: Beispieldatei Übertragungsformat

5 Übertragungsformat in Binärform

Als Alternative zum oben dargestellten JSON-Übertragungsformat wird im Folgenden zur optionalen Benutzung ein Übertragungsformat in Binärform definiert. Dieses Format hat den Vorteil, dass es weniger Datenvolumen bei der Übertragung (oder Speicherung) belegt.

Datensätze im JSON-Übertragungsformat können 1:1 ins binäre Übertragungsformat überführt werden und umgekehrt.

Das Binärformat ist nicht zwingend relevant in Bezug auf eine Zertifizierung, die Umwandlung in das Format erfolgt nach der Signaturbildung, die Rückumwandlung vor der Signaturprüfung und damit sogar vor Aktivwerden der Transparenz-Software.

Das Übertragungsformat besteht aus mehreren Blöcken von Binärdaten.

Zuerst kommen Blöcke, die alle Nutzdatensektionen des JSON-Übertragungsformats abbilden. Danach wird die gebildete Signatur in einem eigenen weiteren Datenblock dargestellt. Außerdem denkbar ist das weitere Anhängen des Public-Keys der Signaturkomponente. Deshalb ist auch der Public-Key als Teil des binären Übertragungsformats umgesetzt.

Das zur Übertragung genutzte strukturierte Binärformat setzt sich aus mehreren Datenabschnitten zusammen: Der erste Abschnitt muss immer das Kopfelement („Header“) sein. Daran schließen sich ein oder mehrere Nutzdatenabschnitte an. Der Header und jeder weitere Datenabschnitt („Chunk“) wird durch einen vorangestellten Bezeichner im ersten Byte des Datenabschnitts eindeutig gekennzeichnet. Der Datenabschnitt mit der Signatur muss stets nach den aus den Variablen resultierenden Datenabschnitten dargestellt werden. Daran kann sich optional auch noch ein weiterer Datenabschnitt für den Public-Key anschließen.

Die Datenabschnitte sollen in der Reihenfolge übertragen werden, in der sie auch in diesem Dokument aufgelistet sind.

Das hier vorliegende Datenformat in Binärform wird in Network-Byte-Order, also im Big-Endian-Format übertragen und dargestellt. Dies gilt insbesondere für Datentypen, die mehr als ein Byte belegen. D.h. das höchstwertige Byte kommt zuerst, das niederwertigste zuletzt im Datenstrom.

Strings werden in diesem Datenformat null-terminiert dargestellt; d.h. zuerst stehen die Bytes des Strings, als ASCII/Latin1 codiert, danach folgt ein Byte mit dem Wert 0, dieses beendet den String.

Innerhalb der Tabellenspalte Key (jeweils ganz rechts) und in Anführungszeichen angegebene Worte beziehen sich auf das Darstellungs- und JSON-Übertragungsformat, welches weiter oben erklärt wurde.

TODO: Übersichtstabelle Datengrößen: mit/ohne Signatur, JSON vs. Binärformat

An die Daten des Binärformats wird die Signatur angehängt. Die Signatur wird allerdings auf der Nutzdatensektion im JSON-Darstellungsformat und nicht basierend auf dem Binärdatensatz berechnet; das heißt, die Signatur wird über dieselben Daten und genauso wie beim JSON-Übertragungsformat gebildet. Damit wird bezweckt, dass ein direkter und damit prüfbarer Bezug zwischen dem evtl. dem Nutzer präsentierten Darstellungsformat und der Signatur besteht.

5.1 StrEnum-Typ

Zur Optimierung von vordefinierten Bezeichnern bei der Darstellung von Strings (Hersteller, Modelle, …) wird eine erweiterter String/Enum-Typ eingeführt mit den Namen StrEnum8 und StrEnum16.

Dieser stellt eine Überlagerung von String und Enum dar.

Primär wird davon ausgegangen, dass es sich bei dem Wert um einen Enum handelt. Für den Wert werden in diesem Fall ein Byte (StrEnum8) oder ein Word (StrEnum16) vorgesehen, es handelt sich also um 8-bit- bzw. 16-bit-Enums. Diese sollten einen ausreichend großen Werte-Raum für mögliche Hersteller von Zählern oder die verschiedenen Modelle eines Herstellers aufspannen.

Stellt das erste Byte (das primär zum Enum gehören würde) allerdings einen Buchstaben oder eine Ziffer dar, ist der gesamte Wert wie ein null-terminierter String (siehe oben) zu interpretieren. Folgende Wertebereiche kennzeichnen einen solchen und werden in diesem Sinne als Buchstaben bzw. Ziffern definiert:

- 48-57: ASCII Ziffern
- 65-89: ASCII Großbuchstaben
- 97-122: ASCII Kleinbuchstaben
- 192-214: Latin1-Buchstaben
- 216-246: Latin1-Buchstaben
- 248-255: Latin1-Buchstaben

Beispiel:

0x01 0x02 – Stellt einen Enum-Wert 0x0102 dar.

0x48 0x61 0x6c 0x6c 0x6f 0x00 – Stellt den String „Hallo“ dar.

5.2 Header

Der Header muss der erste Datenabschnitt im gesamten Datensatz sein.

Der Header identifiziert das Datenformat als solches und beschreibt die Gesamtlänge des Datensatzes. Weiter enthält er die allgemeinen Angaben aus dem JSON-Format.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Header: 0xC0 Open Charging Meter Format	-
3	uint24	Weitere Bytes zur Identifikation: 0xABBECE	-
2	uint16	Länge des gesamten binär Datensatzes einschließlich Signatur, ggf. Public-Key und dieses Headers, in Byte	-
1	uint8	API-Version: Major-Version	FV
1	uint8	API-Version: Minor-Version	
1..*	String	Gateway-Identification	GI
1..*	String	Gateway-Serial	GS
1..*	String	Gateway-Version	GV

Tabelle 8: Datenabschnitt Header

5.3 Paginierung

Dieser Abschnitt stellt die Paginierung dar.

Dieser Nutzdatenabschnitt ist obligatorisch.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Kontext: 0x01	-
1	uint8	Kontext: 0: ohne Transaktionskontext „F“ 1: im Transaktionskontext „T“	PG
4	uint32	Paginierungszähler	

Tabelle 9: Nutzdatenabschnitt Ablesungskontext

5.4 Zähleridentifikation

Dieser Nutzdatenabschnitt identifiziert den Zähler.

Dieser Nutzdatenabschnitt ist obligatorisch.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Zähler: 0x02	-
2..*	StrEnum16	Hersteller des Zählers, Codierung gemäß Tabelle 21 oder Klartext.	MV
2..*	StrEnum16	Modell des Zählers, Codierung gemäß Tabelle 21 oder Klartext.	MM
1..*	String	Seriennummer des Zählers (ASCII)	MS
1..*	String	Firmware-Version des Zählers (ASCII)	MF

Tabelle 10: Nutzdatenabschnitt Zähler

5.5 Benutzerzuordnung

Die Zuordnung des Benutzers erfolgt entsprechend dem Datenschema aus Abschnitt 6.2.

Dieser Datenabschnitt ist genau dann enthalten, wenn Transaktionsbezug besteht. Auch wenn bei Transaktionsbezug noch kein Nutzer zugeordnet werden kann, ist der Abschnitt enthalten. Wenn kein Transaktionsbezug besteht, fehlt dieser Nutzdatenabschnitt.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Nutzerzuordnung: 0x03	-
1	uint8	Nutzerzuordnung: Gesamtstatus und Gruppe aus Tabelle 23 Bit 7: Identification-Status 1=true, 0=false Bit 6-0: Gruppe	IS/IL
1	uint8	Status/Fehler, Wert aus Tabelle 23	IL
2	uint16	Nutzerzuordnung: Details, zusammengesetzt entsprechend Tabelle 24	IF
1	uint8	Nutzerzuordnung: Typ, Wert aus Tabelle 29	IT
1..*	String	Nutzerzuordnung: Daten (ASCII)	ID

Tabelle 11: Nutzdatenabschnitt Nutzerzordnung

5.6 Zuordnung des Ladepunktes

Dieser Nutzdatenabschnitt identifiziert den Ladepunkt.

Dieser Nutzdatenabschnitt ist optional.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Zähler: 0x04	CT
1..*	StrEnum8	Typ der Ladepunkt-Identifikation	
1..*	String	ID des Ladepunktes	CI

Tabelle 12: Nutzdatenabschnitt Ladepunkt

5.7 Eine und mehrere Ablesungen

Ein gesamter Datensatz entsprechend des hier vorgestellten Binärformats kann, genauso wie sein JSON-Pendant, ein oder mehrere Ablesungen enthalten.

Dazu werden im Binärformat einfach Ablesungsabschnitte aneinander gereiht.

Eine neue Ablesung wird stets durch den Abschnitt Ablesung Grunddaten angeführt. Danach können die anderen Abschnittstypen zur Ablesung auftauchen. Ein weiterer Abschnitt mit Grunddaten beginnt die nächste Ablesung.

Beispiel:

1. Ablesung Grunddaten (1. Ablesung)
2. Beschreibung Messwert (1. Ablesung, hier nötig, da Felder nötig)
3. Beschreibung Zählerstatus (1. Ablesung, ebenso wg. Feldern nötig)
4. Ablesung Grunddaten (2. Ablesung)
5. Ablesung Grunddaten (3. Ablesung)
6. Ablesung Grunddaten (4. Ablesung)
7. Beschreibung Zählerstatus (4. Ablesung, nötig weil z.B. Fehler eintritt)
8. …

5.7.1 Ablesung Grunddaten

Dieser Abschnitt stellt die Grunddaten einer Ablesung dar. Dies sind die Zeit und deren Qualität zum Zeitpunkt der Ablesung, außerdem der eigentliche Wert der Ablesung.

Dieser Nutzdatenabschnitt ist obligatorisch.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Ablesung: 0x0A	-

2	uint4 int12	Status der Synchronisation und Zeitzone Hi-Nibble des Hi-Bytes: Synchronisation, Wert siehe Tabelle 31 Restliche Nibbles (12 bit): Zeitzone Die Zeitzone als Dezimalzahl der Form [-]HHMM wird als vorzeichenbehafteter Integer (Zweierkomplement) dargestellt.	TM
6	uint48	UTC Zeit in Millisekunden seit Beginn der Unix-Epoche (1970-01-01T00:00:00,000)	
1	char	Transaction: Bezug der Ablesung zur Transaktion, notiert als Großbuchstabe; Der Wert 0 dieses Bytes kennzeichnet die Absenz dieses Feldes.	TX
1..*	String	Wert der Zählerablesung als String, Form entsprechend JSON-Number-Format	RV
1	uint8	Fehlerhafte Größen, als Bit-Feld, Entsprechung zum Zeichen aus JSON-Format in Klammern: Bit 0 – Energie (E) Bit 1 – Zeit (t) Der Wert 0 dieses Bytes kennzeichnet die Absenz dieses Feldes.	EF

Tabelle 13: Ablesungsdatenabschnitt Grunddaten

5.7.2 Beschreibung des Messwerts

Dieser Abschnitt gibt Information zur Beschaffenheit des abgelesenen Werts.

Dieser Nutzdatenabschnitt ist optional.

Tabelle 14 stellt den Aufbau des Nutzdatenabschnitts dar.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Messwertbeschreibung: 0x0B	-
1..*	StrEnum8	Reading-Identification, Codierung gemäß Tabelle 32 oder Klartext.	RI
1..*	StrEnum8	Reading-Unit, Codierung gemäß Tabelle 33 oder Klartext.	RU
1..*	StrEnum8	**Reading Current Type, Codierung gemäß Tabelle 34: Vordefinierte Stromarten** .	RT

Tabelle 14: Ablesungsdatenabschnitt Messwertbeschreibung

5.7.3 Beschreibung des Zählerstatus

Dieser Abschnitt gibt Information zu Eichrechtlich relevanten Ereignissen am Zähler.

Dieser Nutzdatenabschnitt ist optional.

Tabelle 15 stellt den Aufbau des Nutzdatenabschnitts dar.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Ereignis: 0x0E	-
1	uint8	Status: Wert gemäß Tabelle 22	ST

Tabelle 15: Ablesungsdatenabschnitt Zählerstatus

5.8 Signatur

Dieser Abschnitt umgibt die Rohdaten der digitalen Signatur.

Der Abschnitt ist obligatorisch.

Dieser Block stellt eine Abweichung vom JSON-Datenformat dar, insofern, dass die Signatur-Methode hier und nicht als separater zur Nutzdatensektion assoziierter Block dargestellt ist. Da die Signatur jedoch auf dem JSON-Format gebildet wird, beeinflusst dies nicht die Sicherheit.

Tabelle 16 stellt den grundsätzlichen Aufbau des Datenabschnitts dar.

Länge: (Byte)	Typ:	Beschreibung:	Key:
1	uint8	Id Nutzdatenabschnitt Digitale Signatur: 0xD5	-
1..*	StrEnum8	Auswahl der Signatur-Methode, gemäß Tabelle 35 oder Klartext	SM
2	uint16	Länge des Signatur-Datenblocks in Byte, entsprechend o.g. Tabelle	-
0..*	byte[]	Daten der Signatur	SD

Tabelle 16: Datenabschnitt für Signatur

Bei der Erzeugung der Signatur mit ECDSA und einer Schlüssellänge von n Bit entstehen im Wesentlichen zwei Zahlen mit der jeweils gleichen Bitlänge x. Tools wie openssl speichern diese in einer Binärdatei ab, welche mit ASN.1 strukturiert ist. Zur effizienten Übertragung werden die beiden Zahlen wieder herausgelöst.

Länge: (Byte)	Typ:	Beschreibung:
24	Integer, 192 bit	1. Integer aus der ASN.1-Darstellung der Signatur
24	Integer, 192 bit	2. Integer aus der ASN.1-Darstellung der Signatur

Tabelle 17: Daten der Signatur bei ECDSA, 192 Bit Schlüssellänge

Länge: (Byte)	Typ:	Beschreibung:
32	Integer, 256 bit	1. Integer aus der ASN.1-Darstellung der Signatur
32	Integer, 256 bit	2. Integer aus der ASN.1-Darstellung der Signatur

Tabelle 18: Daten der Signatur bei ECDSA, 256 Bit Schlüssellänge

TODO:

- Signaturen: Nur noch eine Tabelle mit dem grundsätzlichen Aufbau, Längen klar definieren

5.9 Public-Key

Dieser Abschnitt dient optional zur Übermittlung des Public-Key. Er kann durch die Signaturkomponente oder später angehängt werden und ist rein informativ.

Dementsprechend ist der Public-Key derzeit nur im Binärformat abgebildet. Das JSON-Übertragungsformat definiert momentan keinen Weg, einen Public-Key darzustellen.

Tabelle 19 stellt den grundsätzlichen Aufbau des Datenabschnitts dar.

Byte-Offset:	Länge: (Byte)	Typ:	Beschreibung:
0	1	uint8	Id Nutzdatenabschnitt Public-Key: 0xE7
1	1	StrEnum8	Typ des Public-Key, siehe Tabelle 36
2	2	uint16	Länge des Public-Key-Datenblocks in Byte, entsprechend o.g. Tabelle
4	n	byte[]	Daten des Public-Key

Tabelle 19: Datenabschnitt für Public-Key

Länge: (Byte)	Typ:	Beschreibung:
siehe Tabelle 36	byte[]	Binärdarstellung des öffentlichen Schlüssels, dieser ist zweimal so lang, wie durch die Schlüssellänge [bit] angegeben.

Tabelle 20: Daten des Public-Keys bei ECDSA

TODOs:

- RFC5480 Kap 2.2 beachten, ggf. hier einarbeiten

6 Weitere Datenfelder

Aufgrund des Binärformats werden die Felder zu Zustand des Zählers und der effektiven Nutzerzuordnung als gesonderte 16-bit- bzw. 8-bit-Worte dargestellt.

6.1 Zähleridentifikation

6.1.1 Hersteller und Modell des Zählers

Die folgende Tabelle stellt die vordefinierten Werte für die Felder Hersteller und Modell des Zählers für die Identifikation des Zählers im Binärformat dar.

Wert:	Hersteller:	Wert:	Modell:
1	Phoenix Contact	1	EEM-350-D-MCB
2	Inepro Metering	1	Pro 1 Mod
		2	Pro 380 Mod
3	Carlo Gavazzi	1	EM340-DIN.AV2.3.X.S1.PF
		2	EM111-DIN.AV8.1.X.S1.PF
		3	EM210-72D.MV5.3.X.OS.X
4	NZR	1	EcoCount S 85

Tabelle 21: Werte für Hersteller/Modell des Zählers

6.1.2 Zustand / Fehler des Zählers

Dieses Feld wird im Binärfomat als vorzeichenloser 8-bit-Integer dargestellt, es kann gleichzeitig einen Wert annehmen. Die Darstellung als Integer im Binärformat erlaubt eine bessere Flexibilität bei zukünftigen Erweiterungen des Zustands.

Wert:	Kürzel:	Bezeichner:	Beschreibung:
0	N	NOT_PRESENT	Der Zähler ist nicht vorhanden/gefunden worden
1	G	OK	Zähler in Ordnung (Good)
2	T	TIMEOUT	Zeitüberschreitung beim Versuch, den Zähler anzusteuern
3	D	DISCONNECTED	Zähler wurde von der Signaturkomponente getrennt
4	R	NOT_FOUND	Zähler nicht mehr gefunden (ist bereits vorher mind. einmal gefunden worden) (Removed)
5	M	MANIPULATED	Manipulation erkannt
6	X	EXCHANGED	Zähler getauscht (Seriennummer stimmt nicht mehr mit der zuletzt bekannten überein.)
7	I	INCOMPATIBLE	Zähler-Version oder dessen API nicht mit Signaturkomponente kompatibel
8	O	OUT_OF_RANGE	Gelesener Wert außerhalb des Wertebereichs
9	S	SUBSTITUTE	Es wurde ein Ersatzwert gebildet
10	E	OTHER_ERROR	Anderer, nicht näher bekannter Fehler
11	F	READ_ERROR	Zählerregister nicht korrekt ausgelesen; Wert der Auslesung ist nicht gültig

Tabelle 22: Zustand/Fehler des Zählers

6.2 Benutzerzuordnung

6.2.1 Status / Fehler

Dieses Feld kann gleichzeitig nur einen Wert annehmen. Zur Übersichtlichkeit sind die Werte in mehrere Gruppen eingeteilt, eine Auszeichnung der Gruppen erfolgt jedoch nur im Binärformat.

Wert:	Gruppe:	Wert:	Status/Fehler:	Beschreibung:

0	nicht verfügbar	199	-	Das Feld ist nicht angegeben.
1	Status ohne Zuordnung	0	NONE	Es liegt keine Nutzerzuordnung vor. Die weiteren Daten zur Nutzerzuordnung haben keine Aussagekraft.
2	Status mit Zuordnung	1	HEARSAY	Die Zuordnung ist ungesichert; also z.B. durch Auslesen einer RFID-UID.
		2	TRUSTED	Die Zuordnung kann in gewissem Maße vertraut werden, es gibt jedoch keine absolute Verlässlichkeit. Beispiel: Authorisierung durch Backend
		10	VERIFIED	Die Zuordnung wurde durch die Signaturkomponente und spezielle Maßnahmen verifiziert.
		20	CERTIFIED	Die Zuordnung wurde durch die Signaturkomponente verifiziert mit Hilfe einer kryptographischen Signatur die die Zuordnung zertifiziert.
		21	SECURE	Die Zuordnung wurde durch eine sicheres Merkmal hergestellt (z.B. sichere RFID-Karte, ISO15118 mit Plug-and-Charge, etc.)
3	Fehler	200	MISMATCH	Fehler; UIDs passen nicht zueinander.
		201	INVALID	Fehler; Zertifikat nicht korrekt (Prüfung negativ).
		202	OUTDATED	Fehler; referenziertes Trust-Zertifikat abgelaufen.
		203	UNKNOWN	Zertifikat konnte nicht erfolgreich geprüft werden (kein passendes Trust-Zertifikat gefunden).

Tabelle 23: Status- und Fehlerzustände der Nutzerzuordnung

6.2.2 Details

Dieses, im Binärformat als 16-Bit-Wort dargestellte, Feld setzt sich aus vier Sub-Feldern zusammen und gibt Details zum Zustandekommen des Status der Benutzerzuordnung an.

Tabelle 24 beschreibt die Aufteilung in Subfelder, die darauf folgenden Tabellen beschreiben die jeweils gültigen Werte.

Byte:	Nibble:	Inhalt:	Details siehe:
Hi-Byte	Hi-Nibble	Zuordnungsanteil RFID	Tabelle 25
Hi-Byte	Lo-Nibble	Zuordnungsanteil OCPP	Tabelle 26
Lo-Byte	Hi-Nibble	Zuordnungsanteil ISO 15118	Tabelle 27
Lo-Byte	Lo-Nibble	Zuordnungsanteil PLMN	Tabelle 28

Tabelle 24: Aufteilung Details Nutzerzuordnung

Wert:	Bezeichner:	Beschreibung:
0	RFID_NONE	keine Zuordnung per RFID
1	RFID_PLAIN	Zuordnung über externen RFID-Kartenleser
2	RFID_RELATED	Zuordnung über geschützten RFID-Kartenleser
3	RFID_PSK	Ein vorher bekannter gemeinsamer Schlüssel (Pre-shared key) wurde genutzt, z.B. mit einer abgesicherten RFID-Karte.

Tabelle 25: Zuordnungsanteil RFID

Wert:	Bezeichner:	Beschreibung:
0	OCPP_NONE	keine Nutzerzuordnung durch OCPP
1	OCPP_RS	Zuordnung durch OCPP-RemoteStart-Methode

4	OCPP_AUTH	Zuordnung durch OCPP-Authorize-Methode
2	OCPP_RS_TLS	Transport-Layer-Security wurde zur Übertragung der Zuordnung per OCPP-RemoteStart-Methode benutzt.
5	OCPP_AUTH_TLS	Transport-Layer-Security wurde zur Übertragung der Zuordnung per OCPP-Authorize-Methode benutzt.
6	OCPP_CACHE	Zuordnung durch Authorization-Cache von OCPP
7	OCPP_WHITELIST	Zuordnung durch White-List von OCPP
3	OCPP_CERTIFIED	Ein Zertifikat des Backends welches die Nutzerzuordnung zertifiziert wurde eingesetzt.

Tabelle 26: Zuordnungsanteil OCPP

Wert:	Bezeichner:	Beschreibung:
0	ISO15118_NONE	keine Nutzerzuordnung durch ISO15118
1	ISO15118_PNC	Plug & Charge wurde benutzt

Tabelle 27: Zuordnungsanteil ISO 15118

Wert:	Bezeichner:	Beschreibung:
0	PLMN_NONE	keine Zuordnung
1	PLMN_RING	Anruf
2	PLMN_SMS	Kurznachricht

Tabelle 28: Zuordnungsanteil PLMN

6.2.3 Typ

Tabelle 29 beschreibt die möglichen Typen von Nutzerzuordnungen, welche in dem Feld Typ als vorzeichenlose Integer verwendet werden können. Diese Zuordnungen orientieren sich an den Vorgaben aus OCPP. Allerdings sieht OCPP derzeit (Version 1.5) nur 20 Zeichen für das Identifikationsmerkmal vor. Entsprechend der maximalen Länge einer IBAN (34 Zeichen) wurden für den Datenbereichs hier allerdings Zuordnungen bis zu 40 Byte Länge vorgesehen.

Wert:	Bezeichner:	Beschreibung:
0	NONE	Keine Zuordnung verfügbar
1	DENIED	Zuordnung momentan nicht abrufbar (wg. Zwei-Faktor-Autorisierung)
2	UNDEFINED	Typ nicht näher benannt
10	ISO14443	UID einer RFID-Karte entsprechend ISO 14443. Dargestellt als 4 oder 7 Bytes in hexadezimaler Schreibweise.
11	ISO15693	UID einer RFID-Karte entsprechend ISO 15693. Dargestellt als 8 Bytes in hexadezimaler Schreibweise.
20	EMAID	Electro-Mobility-Account-ID entsprechend ISO/IEC 15118 (String mit der Länge 14 oder 15)
21	EVCCID	ID eines Elektrofahrzeugs entsprechend ISO/IEC 15118 (Maximallänge 6 Zeichen)
30	EVCOID	EV-Contract-ID entsprechend DIN 91286.
40	ISO7812	Identification-Card-Format entsprechend ISO/IEC 7812 (Kredit- und Bankkarten, etc.)
50	CARD_TXN_NR	Kartentransaktionsnummer (CardTxNbr) für eine Zahlung mit Kredit- oder Bankkarte, die in einem Terminal am Ladepunkt benutzt wird.

60	CENTRAL	Zentral generierte ID. Kein genaues Format definiert, kann z.B. eine UUID sein. (OCPP 2.0)
61	CENTRAL_1	Zentral generierte ID, z.B. durch Start per SMS. Kein genaues Format definiert. (bis OCPP 1.6)
62	CENTRAL_2	Zentral generierte ID, z.B. durch Start durch Operator. Kein genaues Format definiert. (bis OCPP 1.6)
70	LOCAL	Lokal generierte ID. Kein genaues Format definiert, kann z.B. eine UUID sein. (OCPP 2.0)
71	LOCAL_1	Lokal generierte ID, z.B. intern durch den Ladepunkt erzeugte ID. Kein genaues Format definiert. (bis OCPP 1.6)
72	LOCAL_2	Lokal generierte ID, für sonstige Fälle. Kein genaues Format definiert. (bis OCPP 1.6)
80	PHONE_NUMBER	Internationale Telefonnummer mit führendem „+".
90	KEY_CODE	User-bezogener, privater Key-Code. Kein genaues Format definiert.

Tabelle 29: Typen der Nutzeridentifikation

6.3 Zuordnung des Ladepunktes

Wert:	Bezeichner:	Beschreibung:
0	EVSEID	EVSE-ID
1	CBIDC	Charge-Box-ID und Connector-ID nach OCPP, als Feldtrenner wird ein Leerzeichen benutzt, z.B. „STEVE_01 1".

Tabelle 30: Typen der Ladepunktzuordnung

6.4 Status der Zeit

Wert:	Bezeichner:	Beschreibung:
0	U	unbekannt, unsynchronisiert
1	I	informativ (Info-Uhr)
2	S	synchronisiert
3	R	relative Zeitabrechnung mit einem eichrechtlich akkuraten Zeitgeber, basierend auf einer Info-Uhr. D.h. zu Beginn des Ladevorgangs war eine Zeit mit informativer Qualität vorhanden. Während des Ladevorgangs wurde die Zeit (Dauer) entsprechend der eichrechtlichen Anforderungen erfasst.

Tabelle 31: Status der Zeit

6.5 Beschaffenheit des Messwerts

6.5.1 OBIS-Codes

Um ein Format in Version 1.0 zu definieren, wurden hier verschiedene mögliche OBIS-Codes aufgelistet. Im Nachgang kann noch geklärt werden, welche davon zu verwenden sind. Die Formate aus der Vorgängerversion 0.3 wurden beibehalten.

TODOs:

- Kommentar(e) zu den OBIS-Codes noch analysieren. Dies führt evtl. zu veränderten OBIS-Codes.

Wert:	OBIS-Code:	Beschreibung:
1	1-b:1.8.e	Bezug von elektrischer Energie (Wirkarbeit) aus dem Stromnetz (des Ladepunktbetreibers) an den Kunden.
2	1-b:2.8.e	Lieferung von elektrischer Energie (Wirkarbeit) von dem Kunden an das Stromnetz (des Ladepunktbetreibers).

3	1-b:1.8.0	Bezug von elektrischer Energie (Wirkarbeit) aus dem Stromnetz (des Ladepunktbetreibers) an den Kunden.
4	1-b:2.8.0	Lieferung von elektrischer Energie (Wirkarbeit) von dem Kunden an das Stromnetz (des Ladepunktbetreibers).
5	1-0:1.8.0	Bezug von elektrischer Energie (Wirkarbeit) aus dem Stromnetz (des Ladepunktbetreibers) an den Kunden.
6	1-0:2.8.0	Lieferung von elektrischer Energie (Wirkarbeit) von dem Kunden an das Stromnetz (des Ladepunktbetreibers).

Tabelle 32: Vordefinierte OBIS-Codes

6.5.2 Einheiten

Wert:	Einheit:
1	kWh
2	Wh

Tabelle 33: Vordefinierte Einheiten

6.5.3 Stromart

Wert:	Bezeichner:	Beschreibung:
1	AC	Wechselstrommessung
2	DC	Gleichstrommessung

Tabelle 34: Vordefinierte Stromarten

6.6 Kryptographie

6.6.1 Signatur-Methoden

Wert:	Bezeichner:	Signatur-Algorithmus:	Kurvenname und ggf. Synonyme:	Schlüssel-länge:	Hash-Algorithmus:	Block-Länge:
1	ECDSA-secp192k1-SHA256	ECDSA	secp192k1	192 bit	SHA-256	48
2	ECDSA-secp256k1-SHA256	ECDSA	secp256k1	256 bit	SHA-256	64
3	ECDSA-secp192r1-SHA256	ECDSA	secp192r1	192 bit	SHA-256	48
4	ECDSA-secp256r1-SHA256 (Default in OCMF Version 0.4)	ECDSA	secp256r1, NIST P-256, ANSI X9.62 elliptic curve prime256v1	256 bit	SHA-256	64
5	ECDSA-brainpool256r1-SHA256	ECDSA	brainpool256r1	256 bit	SHA-256	64
6	ECDSA-secp384r1-SHA256	ECDSA	secp384r1, NIST P-384, ANSI X9.62 elliptic curve prime384v1	384 bit	SHA-256	96
7	ECDSA-brainpool384r1-SHA256	ECDSA	brainpool384r1	384 bit	SHA-256	96

Tabelle 35: Vordefinierte Signatur-Algorithmen und ihre Parameter

6.6.2 Public-Key-Typen

Wert:	Bezeichner:	Signatur-Algorithmus:	Kurve:	Schlüssel-länge[1]:	Block-Länge:
1	ECDSA-secp192k1	ECDSA	secp192k1	192 bit	48
2	ECDSA-secp256k1	ECDSA	secp256k1	256 bit	64
3	ECDSA-secp192r1	ECDSA	secp192r1	192 bit	48
4	ECDSA-secp256r1 (Default in OCMF Version 0.4)	ECDSA	secp256r1	256 bit	64
5	ECDSA-brainpool256r1	ECDSA	brainpool256r1	256 bit	64
6	ECDSA-secp384r1	ECDSA	secp384r1	384 bit	96
7	ECDSA-brainpool384r1	ECDSA	brainpool384r1	384 bit	96

Tabelle 36: Vordefinierte Schlüsseltypen

[1] Schlüssellänge des entsprechenden Private-Key