

Distributed Data Validation for a Key-value Store in a Decentralized Electric Vehicle Charging Network

Benedikt Kirpes¹ ^a, Micha Roon² and Christopher Burgahn²

¹University of Mannheim, Germany

²Share&Charge Foundation, Zug, Switzerland

Keywords: E-Mobility Systems, Smart Charging Infrastructure, Information Systems, Distributed Systems, Distributed Data Storage, Distributed Hash Table, Key-value Store, Data Validation, Consensus Mechanism.


Abstract: The mobility sector experiences a fundamental shift to more connected, autonomous, shared and electric means of transportation. For an electric mobility system to function, an efficient and reliable electric vehicle charging network is required. The Open Charging Network, which is built and curated by the Share&Charge Foundation is a digital, open and decentralized infrastructure for operating and connecting assets of the e-mobility ecosystem like charge points and electric vehicles. In such a network validity and consistency of data are crucial. Since the underlying information system is designed based on distributed ledger technologies and distributed hash tables, also the validation of data for the respective key-value store should be implemented and executed in a distributed manner. In this paper, we contribute to the body of research by analyzing the current situation in distributed systems and presenting the design and development of a mechanism for a distributed data validation. We provide an outlook into the future implementation within the Open Charging Network, where the solution will be demonstrated in a suitable context. Further it will be evaluated regarding the primary requirement of data validity and secondary requirements such as availability, reliability and scalability.

1 MOTIVATION

Still many barriers for the widespread adoption of Electric Vehicles (EVs) exist, one of them being the insufficient charging infrastructure (Egbue and Long, 2012). If customer incentives can be set adequately and adoption-hindering barriers are removed, EV sales may increase dramatically, as the example of Norway has shown (Mersky et al., 2016). For a mobility sector that increasingly relies on EVs, the available charging infrastructure constitutes a critical system that must be built, operated and maintained efficiently. This is not only the case for the utilized hardware components, every charging infrastructure also consists of multiple software components. Provision of a software infrastructure for a smart EV charging network with highest availability and security standards is still a big challenge. The information systems research community draws similar conclusions, considering e-mobility as sub-discipline of energy informatics and as a major building block for a sustainable and smart energy future (Kossahl et al., 2012). The devel-

opment of the necessary infrastructure for the Internet has shown that Information and Communication Technologies (ICT) developed, operated and maintained in a distributed and open source manner can be considered as the appropriate way for engineering system-critical infrastructures. Following this vision, the Share&Charge¹ Foundation is building and curating the Open Charging Network (OCN) based on distributed technologies. For use cases like charge point (CP) availability status, reservation, smart routing, roaming and smart charging, such a system requires a shared state of the relevant data, e.g. from smart meters, CPs and EVs. Validating the correctness of data in such a network is very challenging and currently done either on the application layer or based on centralized and non-transparent ruling. Current technologies and systems moreover fail to fully achieve this common shared state based on distributed validation.

Therefore, in this paper we tackle the following research question: *How can the data stored in a distributed key-value store be validated in a distributed*

^a  <https://orcid.org/0000-0002-6583-444X>

¹<https://shareandcharge.com/>

manner in order to establish a common shared state in a decentralized network for EV charging?

To answer this research question, we apply the structured design science research (DSR) methodology as proposed by (Peffer et al., 2008). The first phases are carried out and presented in this paper, while demonstration, evaluation and further iterations are part of future research work. The remainder of this paper is structured according to the DSR process as follows. After providing an introduction to our research in the context of EV charging networks (section 1), we further elaborate on the identified problem and the specific motivation in section 2.1. We define the objectives and requirements for a solution artifact (section 2.2) and give an overview about related work and the state-of-the-art (section 2.3). In section 3 we propose the design for the artifact, give some insights into its development and present an example in the context of the OCN. We conclude the paper in section 4 by discussing our approach, practical implications and its limitations. Further, we provide an outlook into future research activities in this context.

2 DATA VALIDATION IN DISTRIBUTED SYSTEMS

Although the terms distributed and decentralized are often used synonymously, we distinguish them according to the degree of decentralization into three levels: centralized, decentralized and distributed (Figure 1).

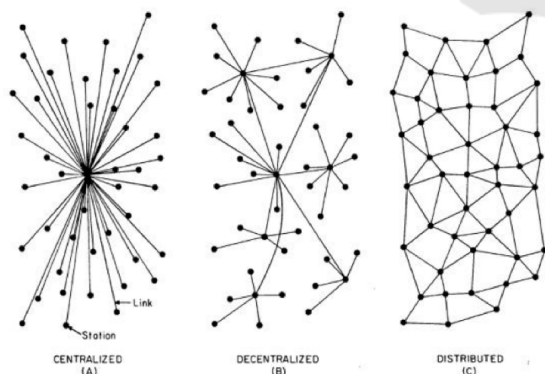


Figure 1: Centralized, Decentralized and Distributed Systems (Raval, 2016).

This classification and terminology apply to technologies, networks and systems alike. Nevertheless, not only distributed, but also decentralized or even centralized systems can be built on top of a distributed technology. Many distributed technologies are avail-

able for building a decentralized network, such as distributed ledger technologies (DLT) like blockchains and directed acyclic graphs (DAG), or distributed hash tables (DHT).

In a distributed or decentralized ICT system, also storage and validation of the respective data should be designed in such a way. To the best of our knowledge, such a distributed validation mechanism has not been proposed or implemented yet. It is not a straightforward task, as data storage itself poses many challenges, e.g. consistency, validity, availability, reliability, scalability, efficiency, integrity, resilience, fault-tolerance, persistence, performance, trust of data and databases; ACID properties (atomicity, consistency, isolation, and durability) for transactions (Decandia et al., 2007).

The implementation and utilization of blockchain technologies to store data in a distributed system only makes sense, if there is a need for trusted public time stamps or a need to prove that the reader always receives the latest state and that there is only one state (Wuest and Gervais, 2018). This is a rather rare case. For all other systems a DHT is a better way (regarding efficiency, performance, scalability) to store data in a distributed way. In a DHT the data gets hashed and stored within a distributed key-value store (KVS), e.g. implemented in Amazon's Dynamo (Decandia et al., 2007) or Apache Cassandra (Lakshman and Malik, 2010). The four initial DHT protocols and algorithms are Chord, CAN, Tapestry, and Pastry. In a DHT, each node in the network holds a portion of the data, unlike for many blockchain technologies where all nodes store the same data. Additionally, all data are stored multiple times for reliability purposes. If one node fails, the data stored in it can be retrieved from one of its peers and the network can recover. The participating nodes coordinate themselves in order to balance and store the data in the distributed network without any central instance.

2.1 Challenges of Distributed Data Validation

In the Open Charging Network by the Share&Charge Foundation, both distributed technologies, blockchain and DHT are utilized, serving different purposes. The blockchain is used as a reference for trusted time stamps. The distributed KVS of the hash table is intended to hold the current state of the system including the state of each CP (e.g. free, occupied) and EV (e.g. plugged-in, charging), the communication information and the ownership details of the assets. Using the integrated blockchain technology for storing this data would have multiple downsides: it would be

inefficient (performance, scalability), overly expensive and modifying the data impossible due to the immutability by design of blockchain technologies. Further, certain personal data would become public, leading to General Data Protection Regulation (GDPR) issues. Hence, the OCN network relies on a distributed KVS which allows for parallel data operations and does not require a network-wide consensus on every transaction. In order to build service applications on top of the system's data the content of the distributed KVS needs to be trusted, also when storing information off-chain.

Major challenges for a distributed KVS are validation and verification of the data, and consequently trust in the data, devices and actors. For *data verification*, it has to be checked and ensured that the data entered into the system is actually the same as the original data, accurate and as intended, e.g. by double entry check. By *data validation*, the data is checked to be reasonable for the system's requirements according to certain rules, e.g. by data format, range or structure check.

In order to make sure that the data stored on each node is valid, the owner of the data element has to sign it. When the reader of the data is an IoT device, it might not have the means to perform validation and might rely on a trusted source to provide it with data. The verification of the signature is a simple process which can be performed by any client, even an IoT device. Besides the verification of the signature, more complex validation and verification of data might be required, which can not be performed by every IoT device, e.g. validations that include complex calculations or access to remote data sources. These kind of validations are very common in database systems where data integrity and other checks are required. In a DHT, data validation rules are typically issued, enforced and maintained by a central authority and not by the distributed network itself.

2.2 Artifact Requirements

In this paper we propose a structured methodology for distributed data validation in a DHT. This would enable decentralized applications (dApps) to store data with complex validation rules in a DHT (as opposed to a blockchain smart contract). Application of such a concept would include several phases. Our resulting artifact will provide means to address these phases and fulfill the following functional requirements:

1. Define validation rules

The artifact must provide a structured methodology to describe the data structure and define validation rules. This should be implemented based

on a (standardized) general purpose or domain-specific language with suitable syntax and notation.

2. Publish validation rules

The artifact must provide a way to enable the creator of the data structure to create and publish the defined validation rules to the network.

3. Agree on validation rules

The artifact must provide a mechanism to achieve a distributed agreement or consensus within the network about the published validation rules.

4. Enforce validation rules

The artifact must provide means to enforce the validation rules and enable all nodes to read and interpret data considering the validation rules.

The generic objectives for data storage as introduced in the beginning of section 2 are extensive and challenging. In our research and for the artifact evaluation, we focus on a relevant and specific sub-set for distributed data validation. This includes the following non-functional requirements: data validity as primary objective; consistency and availability of the data, plus reliability and scalability of the solution as secondary requirements. *Validity* of the data ensures trust in the network. Invalid data of any type may harm the whole system. *Consistency* of data refers to application consistency, transaction consistency and point-in-time consistency (for recovery purpose). *Availability* and *reliability* of the data are critical issues for the smooth operation of the charging network. *Scalability* of the solution is important in order to support a large amount of IoT devices (EVs, CPs, etc).

2.3 State-of-the-Art in Distributed Data Validation

Data validation starts with the definition of the data structure and the specification of a schema. Each data element must comply with the respective schema definition of its namespace. A schema language can be modelled on RelaxNG (ISO/IEC 19757-2, 2003) which has many implementations. LIVR² is a project more focused on validation rules. The Valiktor project³ proposes methods to validate Kotlin objects which could be used to describe the data. Another idea is the use of a content-hash key by a front-end to verify integrity of data received from the network (Sit, 2008). CHAINIAC is a promising approach for publishing of validation rules (Nikitin et al., 2017).

²<https://github.com/koorchik/LIVR>

³<https://github.com/valiktor/valiktor>

The Holochain⁴ DHT embeds the validation rules as a condition for the propagation of data.

Most of the existing approaches focus only on static validation or centralized definition and deployment of validation rules. We will take these concepts one step further and contribute to the body of research by making the validation i) more dynamic and ii) dependent on both the user who submits the data and the content of the data (which Valiktor already is able to do). For example, when an IoT device (in our case an EV or CP) connects to an OCN client⁵ in the network, this information is stored in the global address book which is stored in the DHT. The address book entry must be signed by both the user and a message broker. This means that there will be two signature fields (e.g. `ocn.cli.sig` and `user.sig`) which must contain a signed version of the hash field. When the information is stored by a DHT node it has to find the correct validation rules and only accept data which yields a positive valid result.

3 DESIGN AND DEVELOPMENT OF THE DATA VALIDATION MECHANISM

The artifact to be designed and developed is a concept and mechanism for distributed data validation, applicable for the Open Charging Network and other decentralized networks based on DHTs with a key-value store. Figure 2 visualizes the requirements and the artifact design of our distributed data validation mechanism. The four major components (solution space) cover the four phases of the distributed data validation process (problem space) as stated in section 2.2 to meet the identified functional requirements. Design and implementation of these artifact components are described in the following section.

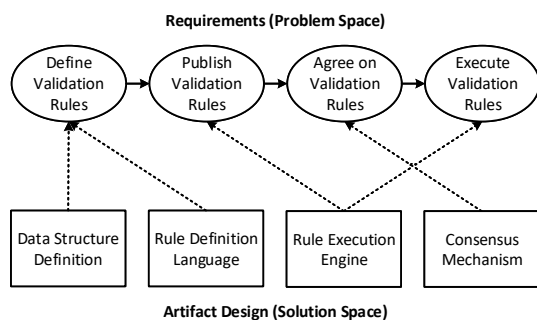


Figure 2: Distributed data validation mechanism.

⁴<https://github.com/holochain/>

⁵<https://bitbucket.org/shareandcharge/ocn-client>

3.1 Data Structure Definition and Rule Definition Language

Prior to defining validation rules, certain assumptions have to be made regarding structure and encoding of the data. Since a DHT is a store of bytes, an encoding standard has to be specified and defined to allow any participant to read and understand the data. A concise language is required for the data structure description, for which inspiration can be drawn from many existing syntax specifications. To the best of our knowledge, no suitable language to describe both, data structure and validation logic, exists. Therefore, we create a specific XML-based grammar in order to provide a complete and specialized language. It will be a subset of an existing Data Definition Language (DDL), similar to those provided by relational database management systems. In its first iteration, the definition of the rules will be done by linking the source code of the rule to the structure described in the DDL. Separating structure and logic will simplify the design.

Namespaces and Domains. A DHT does not have concepts with tables or collections, it only uses keys for referencing data. The data must be attached to a description and this description should not be attached to every key-value pair. A suitable solution is the definition of a namespace schema akin to Java's package naming system. The top-level domain is controlled by the organization, in this case the Share&Charge foundation and is always `.dht`, while the actual domain name is chosen by the user and can be any string conforming to the java package naming conventions. Each domain must be registered with the distributed KVS naming service which records the public keys associated with a specific domain. All namespaces belonging to a domain (namely the sub-domains) are controlled by the top-level public key. In order to guarantee data retention in a distributed KVS, data must be stored multiple times. As not all servers are controlled centrally, there is no way to ensure that a server will stay up and running or that data will not be deleted. The replication ratio will therefore be decided on a namespace level. Some data being more important to the network than others, not all data will be replicated with the same ratio.

Example of a Data Structure Definition and One Instance. In order to illustrate a potential syntax for the data structure definition, we provide a simplified example for charge point data (see above and Appendix). It has been created in eXtensible Markup Language (XML) and the respective XML Schema

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="DataDefinition">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element name="namespace" type="NameSpace"/>
7          <xs:any />
8          <xs:element name="validation" type="ValidationRule" minOccurs=
9            "0" maxOccurs="unbounded"/>
10         <xs:element name="signature" type="xs:string"/>
11       </xs:sequence>
12     </xs:complexType>
13   </xs:element>
14
15   <xs:simpleType name="NameSpace">
16     <xs:restriction base="xs:string">
17       <xs:pattern value="(?:^\w+|\w+\. \w+)+$"/>
18     </xs:restriction>
19   </xs:simpleType>
20
21   <xs:complexType name="DataStructure">
22     </xs:complexType>
23
24   <xs:complexType name="ValidationRule">
25     <xs:sequence>
26       <xs:element name="sourceUrl" type="xs:string"/>
27       <xs:element name="sourceHash" type="xs:string"/>
28       <xs:element name="binaryUrl" type="xs:string"/>
29       <xs:element name="binaryHash" type="xs:string"/>
30     </xs:sequence>
31   </xs:complexType>
32 </xs:schema>

```

Listing 1: Data structure definition (xsd file format).

Definition (XSD). The final implementation will use a different format to be more concise.

3.2 Rule Execution Engine

For the purpose of simplicity, Java is used to create the validation rules. The advantages of Java are its widespread usage with a large amount of capable developers. It can run on almost any hardware and Java bytecode can be transpiled into Webassembly⁶ for native performance on many platforms. Further, many languages compile to Java Virtual Machine (JVM) bytecode and can be used as drop-in replacements, e.g. Jython as a Java implementation for Python. Writing and publishing the validation rules is enabled by an API.

Using the JVM to create validation rules allows usage of the associated execution environment for the actual execution of the validations. Further, it eliminates the need to create a dedicated rules definition language and replaces it with a much simpler API definition. By using an interface-based programming

⁶<https://webassembly.org/>

style (Steimann and Mayer, 2005) we can ensure that provided bytecode can be executed correctly. The process of retrieving the validation rules would be as follows:

1. connect to a blockchain node
2. read the URL for the validation rules' binary required by the namespace
3. fetch the validation rules' binary from the URL
4. verify the binary's hash against the hash stored in the blockchain
5. execute the rules for each data element

3.3 Consensus Mechanism

As mentioned earlier, validation rules have the requirement of completeness. Network participants want to have certainty that they always get the latest version of the validation rule and that a malicious node can not serve outdated rules. Therefore we propose to use a smart contract on the Energy

Web Chain⁷ to reach consensus on validation rules. The consensus mechanism in a distributed system allows to create rules which everyone agrees on and makes it impossible to change the rules without everyone agreeing, or at the very least being informed about a change. The consensus mechanism will be implemented as an Ethereum smart contract which will contain the rules (or an immutable link to them) and allow to reach consensus. The method to proof that the source code associated with a particular validation rule is correct is the following:

1. push the source code for the validation rule to a git repository
2. push the binary bundle to the same repository
3. publish a transaction to the rules smart contract with:
 - the namespace for which the rule applies to (the sender must be the owner of the namespace)
 - the URL to the git repository (must be publicly reachable)
 - the commit hash of the published version of the code
 - the URL to the binary bundle
 - the hash of the binary bundle

The validation rules are only accepted, once a pre-defined number of DHT nodes has verified that the published source code yields the published binary. The verifying nodes are selected by using Scalable Bias-Resistant Distributed Randomness (Syta et al., 2017) to ensure that a malicious actor can not validate its own code.

The validation mechanism is based on a standardized build mechanism which is easy to implement in most JVM languages. In order to make the build process tamper-proof, the Maven or Gradle scripts for building rules will not be provided by the validation rule developer but by the DHT node verifying the code. It is the developer's responsibility to make sure the build process works. Maven and Gradle scripts for Java, Kotlin and Scala will be provided in the first version. The proposed mechanism with its four components is in development and as a next step will be fully implemented, demonstrated and evaluated within the Share&Charge solution.

3.4 Demonstration Example in the Open Charging Network

In the OCN developed by Share&Charge the KVS will be used to record the state of the available

⁷<https://www.energyweb.org/technology/energy-web-chain/>

assets. A concrete example for that is the state of the charge points. According to the specification of the widely used peer-to-peer *Open Charge Point Interface (OCPI)* (NKL, 2019), a charge point can be in one of 9 possible states: AVAILABLE, BLOCKED, CHARGING, INOPERATIVE, OUTFORDER, PLANNED, REMOVED, RESERVED, UNKNOWN (see Appendix).

When an EV driver is looking for a charge point to recharge the battery the AVAILABLE status of the charge point is a necessity. This information about the state of the charge point is with the Charge Point Operator (CPO). The driver typically gets information from an e-mobility service provider, which aggregates charge points from many CPOs. Thus, the easiest way to ensure that the driver gets accurate information about charge points from any CPO in the network is to store it in a shared data-store. Creating a dedicated KVS for charge point status data would save the e-mobility service providers many queries for the state of the charge points.

Validation of Charge Point States. It is important to ensure that the provided information originates from the correct source and valid. Else the system could be harmed, e.g. if a malicious CPO would be able to set the state of all competitor charge points to BLOCKED in order to hurt their business. The namespace `com.shareandcharge.ocpi.location` (which contains all the charge points) must be owned by the independent Share&Charge Foundation in order to allow all CPOs to contribute and provide a single source for location data. This means that certain validation rules would need to be enforced:

- Each location defines its owner
- The owner must sign each entry (creation and update) with its private key
- The ID of each location must be unique throughout the entire network

This ensures that once a CPO has published a location (charge point) it is the only one being able to update its state.

4 CONCLUSION AND OUTLOOK

A resilient EV charging infrastructure based on distributed technologies requires a DHT to share information about assets and their state. A trustworthy distributed data validation is a key element for such a system being able to rely on the data input that is provided by the actors of it. Today, static and centralized data validation in a DHT already exist. However,

as an open EV charging network requires highly dynamic data input like real-time availability of CPs, a more dynamic and distributed data validation mechanism is required. Linked to the functional requirements of a distributed data validation process, we outlined the design and development of four artifact components: Data Structure Definition, Rule Definition Language, Rule Execution Engine and Consensus Mechanisms.

Limitations, Future Research and Practical Implications.

One of the major challenges in building a decentralized charging network including a DHT is to gain the necessary adoption. The business decisions that would drive such an adoption are linked to an agreement on the rules that a data validation process should follow. Defining the right rules that all users in a certain application market can agree upon can be a very complex task to do, which was not discussed further in this paper. However, the research conducted can be used for providing the technical means to implement such rules. The Share&Charge Foundation is focused on facilitating this decision making process in the EV charging industry and will conduct further research into the outcomes of this process. For future research, we plan to extend on this research activity by demonstrating the usefulness of the implemented mechanisms in an appropriate system context. The concept will be applied for the Open Charging Network. The developed and implemented artifact will be evaluated against the identified requirements from this paper.

ACKNOWLEDGEMENTS

We thank Dietrich Stümmermann, Adam Staveley and all other employees of the Share&Charge Foundation for their valuable input and discussions. Part of this research was funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 713864.

REFERENCES

- Decandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's Highly Available Key-value Store. In *SOSP'07*.
- Egbue, O. and Long, S. (2012). Barriers to widespread adoption of electric vehicles: An analysis of consumer attitudes and perceptions. *Energy Policy*, 48(2012):717–729.
- ISO/IEC 19757-2 (2003). Document Schema Definition Language (DSDL) Part 2: Regular-grammar-based validation (RELAX NG).
- Kossahl, J., Busse, S., and Kolbe, L. (2012). The Evolution of Energy Informatics in the Information Systems Community - A Literature Analysis and Research Agenda. In *20th ECIS Proceedings*.
- Lakshman, A. and Malik, P. (2010). Cassandra. *ACM SIGOPS Operating Systems Review*, 44(2):35.
- Mersky, A. C., Sprei, F., Samaras, C., and Qian, Z. S. (2016). Effectiveness of incentives on electric vehicle adoption in Norway. *Transportation Research Part D: Transport and Environment*, 46:56–68.
- Nikitin, K., Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Khoffi, I., Cappos, J., and Ford, B. (2017). CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds. In *26th USENIX Security Symposium*, pages 1271–1287.
- NKL (2019). Open Charge Point Interface (OCPI) 2.2. Technical report, Netherlands Knowledge Platform for Public Charging Infrastructure (NKL).
- Peffer, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77.
- Raval, S. (2016). *Decentralized Applications - Harnessing Bitcoin's Blockchain Technology*. O'Reilly Media, Inc.
- Sit, E. (2008). *Storing and Managing Data in a Distributed Hash Table*. PhD thesis, Massachusetts Institute of Technology.
- Steimann, F. and Mayer, P. (2005). Patterns of interface-based programming. *Journal of Object Technology*, 4(5):75–94.
- Syta, E., Jovanovic, P., Kogias, E. K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M. J., and Ford, B. (2017). Scalable Bias-Resistant Distributed Randomness. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 444–460.
- Wuest, K. and Gervais, A. (2018). Do you need a blockchain? In *Crypto Valley Conference on Blockchain Technology, CVCBT 2018*, pages 45–54.

APPENDIX

```

1 <sc:DataDefinition xmlns:sc="file://data-def.xsd" xmlns="http://www.w3.org/2001/XMLSchema">
2   <sc:namespace>name</sc:namespace>
3   <sc:structure xmlns:ds="http://www.w3.org/2001/XMLSchema">
4     <ds:element name="Location">
5       <ds:complexType>
6         <sequence>
7           <ds:element name="id" type="ds:string"/>
8           <ds:element name="name" type="ds:string"/>
9           <ds:element name="state">
10            <ds:complexType>
11              <choice>
12                <ds:element name="AVAILABLE"/>
13                <ds:element name="BLOCKED"/>
14                <ds:element name="CHARGING"/>
15                <ds:element name="INOPERATIVE"/>
16                <ds:element name="OUTOFORDER"/>
17                <ds:element name="PLANNED"/>
18                <ds:element name="REMOVED"/>
19                <ds:element name="RESERVED"/>
20                <ds:element name="UNKNOWN"/>
21              </choice>
22            </ds:complexType>
23          </ds:element>
24          <ds:element name="gps-coordinates">
25            <ds:simpleType>
26              <ds:annotation>
27                <ds:documentation>
28                  Valid GPS coordinates in the form of lat/long
29                </ds:documentation>
30              </ds:annotation>
31              <ds:restriction base="ds:string">
32                <ds:pattern value="^( [+]? ) ( [ \d ] { 1, 2 } ) ( ( ( \. ) ( \d + ) ) ( \s * ) ( [ - + ] ? ) ( [ \d ] { 1, 3 } ) ( ( \. ) ( \d + ) ) ? ) $" />
33              </ds:restriction>
34            </ds:simpleType>
35          </ds:element>
36          <ds:element name="owner">
37            <ds:simpleType>
38              <ds:annotation>
39                <ds:documentation>
40                  CPO who owns the charging location. Must be a valid Ethereum address.
41                </ds:documentation>
42              </ds:annotation>
43              <ds:restriction base="ds:string">
44                <ds:pattern value="^0x[a-fA-F0-9]{40}$" />
45              </ds:restriction>
46            </ds:simpleType>
47          </ds:element>
48        </sequence>
49      </ds:complexType>
50    </ds:element>
51  </sc:structure>
52  <sc:validation>
53    <sc:sourceUri>https://github.com/path-to-source-bundle.zip</sc:sourceUri>
54    <sc:sourceHash>b12c5a41ac005664dc94c0658148eb99</sc:sourceHash>
55    <sc:binaryUri>https://github.com/path-to-binary-bundle.zip</sc:binaryUri>
56    <sc:binaryHash>8ec10e91a9731c337582c79a69506132</sc:binaryHash>
57  </sc:validation>
58  <sc:signature>4d6348865f1713f316ccab4f941bdbf0f93bc66b25f39e1f6b5e2f8510c2cd514d6348 865
59    f1713f316ccab4f941bdbf0f93bc66b25f39e1f6b5e2f8510c2cd51A0</sc:signature>
60 </sc:DataDefinition>

```

Data definition example (xml file format)