



A circular economy approach for life cycles of products and services

Development Report and Documentation for Traceability Components and Tools

Deliverable 5.2

PROJECT INFORMATION	
Type of Project	European Commission Horizon 2020
Topic	CIRC-01-2016-2017 Systemic, eco-innovative approaches for the circular economy: large-scale demonstration projects
Grant Agreement No.	776503
Project Duration	01/05/2018 – 30/04/2021 (36 months)
Project Coordinator	Nottingham Trent University (NTU)
Project Partners	Enviro Data (ENV), Jonathan Michael Smith (JS), Kosnic Lighting Limited (KOS), Centre of Research for Energy Resources and Consumption (CIR), European EPC Competence Center GmbH (EECC), The Institute for Ecology of Industrial Areas (IETU), SWEREA IVF AB (SWE), Make Mothers Matter (MMM), ONA PRODUCT (ONA), INDUMETAL Recycling (IND), GS1 Germany GMBH (GS1G), Laurea University of Applied Science (LAU), Center for European Policy Studies (CEPS), Institute of Communication and Computer Systems (ICCS), Recyclia (REC), S.A.T. Alia (ALIA)
DOCUMENT INFORMATION	



Title	Development report and documentation for traceability components and toolsTest
Version	1.0
Release Date (dd/mm/yy)	2019-07-30
Work Package	WP5
Dissemination Level	PU

DOCUMENT AUTHORS AND AUTHORISATION

Document Responsible	Dr. Sebastian Schmittner, EECC
Contributors	Dr. Sebastian Schmittner, EECC; Dr. Georg Schwering, EECC
Reviewed by	Dr. Bahar Cat-Krause, GS1G; Dr. Fernando Círez Oto, CIR
Approved by	Prof. Daizhong Su, NTU

DOCUMENT HISTORY

Version	Date (dd/mm/yy)	Description	Implemented by
0.1	29/05/19	First draft	EECC
0.2	12/07/19	Draft for Review	EECC
0.3	23/07/19	Reviewers minor corrections included	EECC
0.4	25/07/19	Including appendix, summary and taking all suggestions by reviewers into account	EECC
1.0	26/07/19	Final Draft for Approval	EECC



Summary

In this document, the EECC, as the leader of work package (WP) 5 within CIRC4Life, reports on the development of traceability tools and core components, collectively referred to as the Traceability Module, which is the primary objective of this work package. A “tool” here concretely refers to a RESTful web services (RWS) by which dynamic data about assets (mostly trade items) is captured or shared. Data is called “dynamic” if it is specific to a single instance of the asset, or at least to a lot or batch. Furthermore, dynamic data is not known a priori for the whole product class (such as e.g. the weight of the product) but results from events happening to the concrete assets, such as e.g. transportation of an item. The role of the Traceability Module within the ICT architecture of CIRC4Life is to capture such dynamic data from partners along the value chain, store it, and make it accessible to the other components of the ICT platform.

A major goal of CIRC4Life is to develop an online assessment of the ecological and/or social impact of an individual product. This means that, for example, a customer is enabled to get an ecological score of a product in a shop on demand through an online application and on an item specific level. This enables the customer to compare e.g. tablets of the same kind in terms of their possibly different impacts. This way, new business models are enabled, since it becomes possible to e.g. demand higher prices for items which were manufactured and handled according to higher ecological and social standards and for which accurate dynamic data is available to prove this. Once dynamic data is tracked in the way envisioned in CIRC4Life, also manufacturers, merchants, carriers, and recyclers along the value chain can be provided with a detailed view on the impacts they are causing, which add up to the score of the items being handled. Beyond raising awareness, this information can be taken into account also for the price in B2B trade. This way, an economic incentive is created to reduce ecological and social impacts. At the same time, the tools developed here enable the involved companies to identify which processes have the biggest potential for impact reduction.

The Traceability Module plays the role of the source of dynamic data to the rest of the ICT in CIRC4Life. By developing the tools to capture, store, and share the dynamic data among all involved partners, WP 5 lays the foundation for the online computation of dynamic scores. This first step can be broken down further and multiple tools and components have been developed to fulfil the task. In a first step, the companies along the value chain of a product need to record the relevant data, such as energy or more general resource consumption, waste production, etc. This data needs to be collected and transformed in order to make it accessible and usable. To this end, EECC has developed capturing applications, which are adapted to the digitalization level of the respective partners. Where data is not yet digitized, the EECC provides web user interfaces to enter the data manually. Where data is already available in digital form, file uploads or APIs (RWS) to directly connect to the partners ICT systems are developed.

EECC has developed an extension to the EPCIS 1.2 standard for capturing impacts along with standard business event data in the traceability core component. Building upon the EPCIS standard [\[3\]](#) ensures that the data format and the standard EPCIS tools (SOAP endpoints) can be used beyond this project without further documentation. The development of all core components of the Traceability Module is finished and a test installation has been deployed. EECC has further developed a data access model and protocol. The result is contained in this document. In CIRC4Life, the ICT system is centralised, but in a more distributed setup, the access model enables all partners to retain full control over their data and access rights while at the same time enabling the sharing of the relevant information about ecological and social impacts.

Specific tools to capture and share the data for the recycling and reuse business model and the respective demonstrations have been developed, implemented, and a test installation is deployed. EECC has also developed tools for the other business models, but since these are still in a more conceptual phase and specifications of concrete demonstrations have been developed only partially, those tools are not fully implemented and deployed. The discussion between EECC and all business models and demonstrations is ongoing. EECC has postponed planned resources to finish the implementation and deploy the tools as soon as



the planning in the remaining business models and demonstrations is advanced enough to derive their requirements more concretely.

All traceability tools are documented using open API specifications [\[6\]](#) (OAS, formerly known as Swagger specifications). This means that the documentation is built into the tools and available in machine readable as well as human readable form directly from the software. The latest version at the time of writing can also be found in [Appendix 2](#).



Table of Content

Summary	2
Table of Content	4
List of Tables	6
List of Figures	6
Acronyms and abbreviations	7
Introduction and Goals	8
Dynamic Eco Scores	9
Ecological Transparency and Data Ownership Along the Supply Chain	10
Traceability all along the value cycle	10
User Stories	11
Meat Production	11
Vegetables Production	11
Electronics Recycling and Reuse	12
Food Waste Recycling	14
Solution Strategy	14
System Scope and Context	15
Dynamic Data Sources	16
Data Protection and Privacy	18
Building Block View	19
Design and Development of the Core Components of the Traceability Module	20
Design and Development of Web Services	21
Capturing Services	22
Simplified Impact Capturing	22
Usage and Production Capturing APIs	23
Farming UI	23
Recycling and Reuse Capturing Application	24
Recycling Evaluation Capturing Web Interface	25
Accessing Services	26
Dynamic Eco Data	26
Item Status	26
Bin Filling Overview	26



Design and Development of Core Features of the Traceability Module	27
Data Access Model	28
Named Information AKA Hash URIs	28
Security Gained by Using Named Information	29
Thread Model	29
Solution	30
Double Envelope	30
Comparison to the State of the Art	31
Examples	32
Receiving an item	32
Hiding More Complex Information	33
The EPCIS Extension for Ecological Impacts	33
XML Schema Definition (XSD)	33
Examples for Events in the Demonstration of the Recycling/Reuse CEBM	39
Bin Disposal	39
Bin Collection	40
Inspection Event	41
Repairing Event	42
Disassembly Event	43
Acknowledgment	44
References	44
Appendix	45



List of Tables

[Table 1: Abbreviations](#)

List of Figures

[Figure 1: Material and product flow in a universal value chain.](#)

[Figure 2: Sequence diagram for disposal of electronic devices at a smart bin.](#)

[Figure 3: Sequence diagram for information query of electronic devices disposed at a smart bin.](#)

[Figure 4: Sequence diagram for Inspection of electronic devices disposed at a smart bin at IND's facilities.](#)

[Figure 5: High level data flow diagram of the Traceability Module.](#)

[Figure 6: General value circle for CIRC4Life demos](#)

[Figure 7: Simplified data flow form the electronics recycling and reuse demo to the Traceability Module.](#)

[Figure 8: Data flow diagram through the Traceability Module.](#)

[Figure 10: Components of the EPCIS core system of the Traceability Module](#)

[Figure 11: Generic Impact Capturing Web UI](#)

[Figure 12: Farming Web User Interface Wireframe](#)

[Figure 13: GUI for Inspection of disposed EEE items at IND](#)

[Figure 14: Map like UI for collection tour planning](#)



Acronyms and abbreviations

Abbreviation	Description
AKA	Also Known As
API	Application Programming Interface
B2B	Business to Business
CEBM	Circular Economy Business Model
D X.Y	Deliverable X.Y, referring to a public deliverable within CIRC4Life.
EPC	Electronic Product Code - see [5]
EPCIS	Electronic Product Code Information Services - see [3]
GDPR	General Data Protection Regulation
(S)GLN	(Serialised) Global Location Number
(S)GTIN	(Serialised) Global Trade Item Number
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NI	Named Information
OAS	OpenAPI Specification (originally known as the Swagger Specification)
PU	Public, fully open, e.g. web
REST	Representational State Transfer
RWS	RESTful Web Service
SaaS	Software as a Service
T X.Y	Task X.Y within CIRC4Life
WP X	Work Package X within CIRC4Life

Table 1: Abbreviations

Introduction and Goals

Traceability traditionally refers to capturing data about what happens to the important assets. By this means, it is possible to keep track of the full history and in particular the latest location, disposition, etc. of e.g. consumer goods along the supply chain up to the point of sales. Within CIRC4Life, the overarching goals are twofold. On the one hand, the EECC has developed means to keep track of additional information needed for a life cycle assessment or other ecological scores such as the CO₂ footprint in a standardized form. Using standards makes it possible to share the recorded data between different companies, which is crucial in order to get a reasonably complete picture of the ecological impacts along the whole supply chain. On the other hand, it is to be demonstrated in CIRC4Life how to trace a product beyond the point of sales by recording data also from the usage phase and even from the end of life/reuse/recycling phase. In consequence the loop is closed by demonstrating how materials flows can be traced and hence ecological impacts assessed throughout the full value cycle.

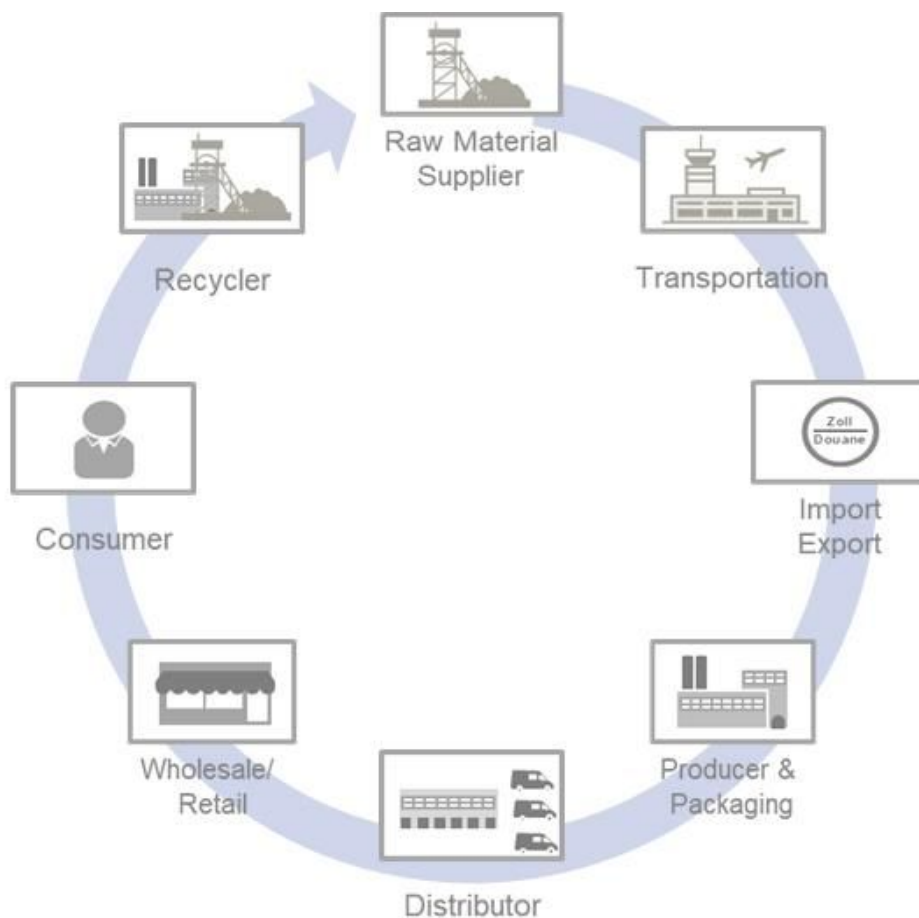


Figure 1: Material and product flow in a universal value chain.

Based on the grant proposal, discussions with the partners within CIRC4Life, and the System specification and requirements analysis (D5.1), overarching goals were combined with the following goals of CIRC4Life to lay the basis for the development of the Traceability Module (T 5.2).



Dynamic Eco Scores

A major innovation developed within CIRC4Life is the dynamic computation of Life Cycle Assessments (LCA) for individual products and the derived eco points (eco credit/debit) scores. Dynamic here means that life cycle assessment, or more generally ecological scoring methods, are improved such that a close to real time computation of a rating of the ecological impact is possible. Furthermore, such impacts can now be computed for individual products such that also very individual products as produced e.g. in the industry 4.0 can be rated. Apart from the obvious benefit of an automated computation being timely available, the qualitative improvement of a score being specific to the individual item should not be underestimated. Primary data, collected by tracing an individual item, can remove the statistical uncertainty from the scoring that is usually introduced by resorting to average values. Even if the relevant figures are collected with questionnaires from the companies involved, as is the usual procedure in LCA assessments, these are still estimated averages. For example, a company probably knows the average annual power consumption of a factory, but measuring this figure with automated sensors and processing the data digitally makes it possible to use daily measurements and hence get a much more accurate estimate of the energy used per produced item. In reality, data gaps remain. Not all companies have the sensors or tools needed or are willing to introduce processes to measure the data relevant for the assessment on a fine grained time scale. In this case, data gaps still need to be filled with average data. But highlighting the data quality and uncertainties in an ecological scoring gives a much more reliable final result. Furthermore, data quality of ecological data can be made a new quality criterion. Just making the average data quality used in the scoring transparent and highlighting the importance of reliable data poses an incentive for the companies involved to gather more reliable data. This process by itself will lead to improved awareness and a focus on ecological impacts while at the same time providing an empiric basis for improving a company's ecological performance.

The dynamic ecological scores boost all CEBMs developed within CIRC4Life. In particular, they enable providing incentives for collaborative recycling and reuse by making the ecological impact of these actions transparent and accessible. Certificates for particularly eco friendly designed products or even just the transparency of making eco scores of products public provides an incentive for the creation of particularly eco friendly products and services on the business side and for sustainable consumption on the consumer side. Additionally, the manufacturing industry is sensitized to ecological impacts of their processes by collecting the data in the first place. Demanding high data quality pushes the need to collect ecologically relevant data up the supply chain.

To achieve these goals, developing the dynamic scoring methods for the computation is important. Before making the scores computable and demonstrating the whole concept, traceability tools are needed in order to capture the data as a basis for input. In particular, the ecological impacts related to each individual item along its supply chain and further on through its whole life cycle need to be traced.

Consequently, the main goal of the Traceability Platform developed in WP5 is to support this major goal of CIRC4Life, i.e. to enable the dynamic computation of eco scores by tracing product specific data of the ecological impacts down to individual items. To make this goal more tangible, EECC has developed user stories (see in particular Section "[Meat Production](#)") together with the partners involved in the demonstrations.

Capturing, Storing, and providing access to such data about individual items on a very fine grained level is the original purpose of traceability tools in general. Within CIRC4Life, the role of EECC is to provide a fully EPCIS 1.2 standard [\[3\]](#) compliant Traceability Module that fulfils this role. By using standards, it is ensured that the developed system is interoperable, usable, and fit for integration beyond CIRC4Life. However, the EPCIS is extensible which is used in order to introduce means for also tracing ecological impacts. The details of this extension are described in Section [The EPCIS Extension for Ecological Impacts](#).



Ecological Transparency and Data Ownership Along the Supply Chain

Goods are traced along the supply chain, but, as of today, information flow from one partner to the other is usually minimal. In a supply chain with many partners, it is rare that the second next company in the chain will get any information above the legal and functional minimum required level. In order to enable the [Dynamic Eco Scores](#), it is essential that data for the ecological impacts for a specific product are shared along the whole supply chain.

In CIRC4Life, there is a central Traceability Module and a central ICT platform, hence partners inherently give the control over their data out of hands by sharing it with the platform. In order to carefully control the data access and guarantee a reasonable level of data protection, the ICT platform implements an access layer that provides single sign on and platform wide roles for users and applications. The Traceability Module contains a client to utilise and obey the identification, authentication and authorisation provided by the ICT platform. This is very suitable for means of all demonstrations implemented in CIRC4Life.

Nevertheless, demanding that companies along the supply chain share their data in this way poses a barrier for adoption of the software developed in CIRC4Life beyond the project itself. Mitigating this foreseeable risk is identified as an important goal of EECC's efforts. EECC has therefore developed a concept for a data access model which enables all companies to maintain full control of their data and at the same time enables partially sharing the relevant (in this case ecological) data in a very fine grained way. Utilising these ideas, confidential information (such as production volumes) can be kept private, whilst it is still possible to assess the total ecological impact of a product. Since this concept involves a distributed system, the full implementation is beyond the scope of CIRC4Life. Nevertheless, EECC has designed a future proof data model in such a way as to enable implementation the full system. The details are given in Section [Data Access Model](#).

Traceability all along the value cycle

Traceability systems that are in operation today typically trace a products life cycle only up to the point of sale. Within CIRC4Life, it is to be demonstrated how to trace an item along its whole value cycle, from production and usage to its end of life. EECC has developed the data models and capture interfaces needed to collect data from the meat and vegetables as well as the electronics supply chains. Ecological impacts, such as the total energy consumption and maintenance related data of lamps during the usage phase will be captured along with the usual traceability data in order to support the leasing model of KOS which demonstrates a sustainable consumption business model.

The Traceability Module also facilitates the data exchange between the intelligent bins tested within CIRC4Life and the rest of the ICT Platform. Subsequent inspection, possible remanufacturing, and reuse or recycling of items is traced as well. This means that what happens to consumer electronics is traced even beyond the end of life. In order to demonstrate that the system is generically usable, it is also to be used in order to trace what happens to the bio waste after collection, which is part of Task 6.5 (demonstration of CEBMs for the meat supply chain). In both demonstrations, secondary raw materials are produced from the recycling. The compost or secondary metals and plastics then re-enter in the production stage hence closing the cycle. Overall, utilising traceability along the whole value cycle of circular economy will be demonstrated and circular economy business models can benefit from the sharing of item specific data about ecological impacts.



User Stories

Meat Production

Traceability systems based on EPCIS for meat and meat products are already established in many businesses. The most common use case is to fulfil the regulatory requirements of providing information about the origin of the meat. In today's traceability systems, however, information on the sustainability of the production is usually not captured, much less is it carried along with the product. This means that the basic data for the computation of [Dynamic Eco Scores](#) is missing.

In the demonstration, the end user comes to a shop and is provided with additional information about the meat products on offer. A label featuring an easy to understand visualisation of the eco debit of the product is attached to each item. Additionally, there is a digital link in the form of e.g. a QR code, which can be scanned with the CIRC4Life eco shopping application, developed by NTU. After scanning the label that includes a serial number, the individual eco debit of exactly this single item is shown along with the possibility to obtain more detailed data of the product. This allows the end user to conveniently access the available static information of the product (manufacturer, weight, etc.), the lot based information, such as the best before date, as well as item specific information. The eco debit computation will be as specific as possible.

Another user story is developed by ALIA as a part of their business model demonstration. The key objective here is to improve the ecological performance of the meat supply chain. For the responsible manager it is important to get a process and maybe time specific view on the eco impacts in order to identify the processes that have the biggest impact and hence also the biggest potential for improvement.

The Traceability Module offers all means to collect and provide the data along the meat supply chain. EECC has developed the corresponding EPCIS event data models which can in principal track and trace all ecological impacts on the most fine grained level.

In reality, it is usually difficult to trace a piece of meat back to an individual animal. The intermediate “product” of the rearing, which is slaughtered for meat production, might be individually tracked through (RFID enabled) ear tags. However, the batch size at the slaughterhouse usually prevents tracing a meat product back to an individual farm. In this case, sustainability information can only be provided on a batch/lot level and not on the level of the individual item. This is, however, much more accurate than the current state of the art which only takes information on product class level into account for LCA. By capturing life data from ALIA's systems throughout the supply chain, the Traceability Module will be able to provide the data to enable close to real time feedback on changes in the eco debit of the currently produced products upon modification of the processes involved. Hence ALIA is enabled to dynamically adapt and improve their production in order to become even more eco-friendly.

Vegetables Production

CIRC4Life consortium member Jonathan Smith (JS) already collects data to identify ecological impacts of the vegetable production of his small organic farm business. In particular, he has focussed on carbon dioxide emissions and collected data on a yearly basis so far. With the traceability tools, he can collect data more timely and more specifically and hence he is enabled to break the ecological impact of his farm down to the individual products. The case of comparing e.g. one euro worth of salad with one euro worth of potatoes in terms of their CO₂ footprint was identified to be a relevant KPI. Given the digitalisation level of his business, EECC has decided to develop a graphical user interface for JS to capture the traceability data in regular time intervals manually.

Furthermore, the online computation methods developed within CIRC4Life (see [Dynamic Eco Scores](#)) can provide him with an up to date view of the current ecological performance of the individual parts of his farm.

This way, he can adapt much more quickly and evolve his business towards a more eco-friendly production on a monthly instead of a yearly time scale.

Providing the information on the ecological debt associated with JS's products to the end customer was identified to be less relevant in this case, since he has a personal relation to most of his customers and they are quite well informed about his ecological efforts, anyway. Nevertheless, providing quantitative comparison of his ecological performance, in particular in direct comparison to the conventionally produced vegetables on offer in supermarkets, will improve his credibility through transparency.

Electronics Recycling and Reuse

A major development of CIRC4Life is the business model of collecting small electronic devices, such as tablets and mobile phones, from the end user for reuse and recycling. To facilitate the collection, smart bins are set up in central public locations and schools. IND will collect the electronic waste in the demonstration of this use case. The devices are then assessed in terms of reusability. They may be repaired/re-manufactured and then sold at low prices to institutions which do not demand the latest cutting edge technology devices, in particular to schools. If the devices are found to be broken, they are recycled. Even if the project only achieves an increased recycling rate of electronic devices, this will be a success, given the currently very low rates. For further details of the business model underlying this use case, please refer to Deliverable 2.1.

A sequence diagram that shows all steps of this demonstration that are to be supported by the Traceability Platform is shown in the Figures below. First an end user decides to recycle one of his devices using the CIRC4Life tools (see [Figure 2](#)). The customer can identify himself to the smart bin using a QR Code scan from the Recycling and Reuse mobile application developed by NTU. The bin then prints a label with a barcode to be attached on the item before throwing it into the bin. In the background, the application and the bin send the relevant information to the Traceability system which can hence set up tracking of the recycled item, or continue the track if the item is already known to CIRC4Life, because it was bought using the CIRC4Life eco shopping application.

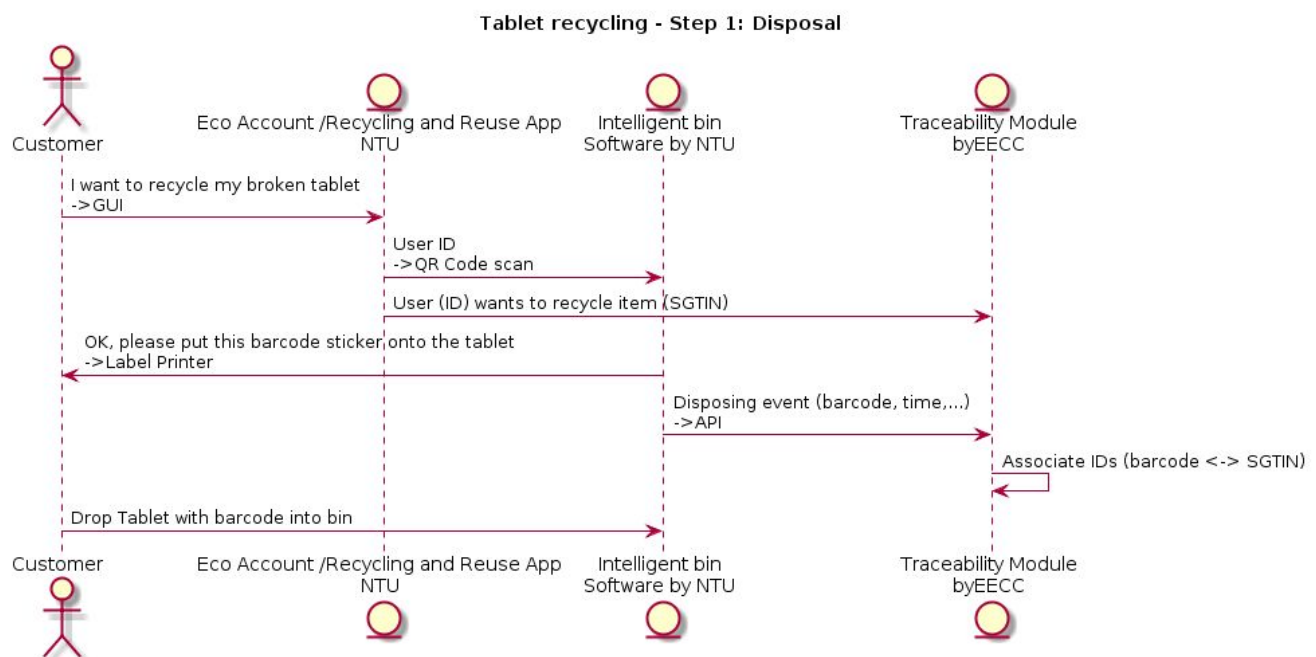


Figure 2: Sequence diagram for disposal of electronic devices at a smart bin.

The latest status of any item being tracked by the Traceability Module can be queried using the web service endpoint, which EECC developed specifically for this purpose. A lightweight caching database is used on top of the Traceability Module core in order to provide very fast response times here, since this is a crucial feature for the user experience. See [Figure 3](#) for a depiction of the sequence of interactions.

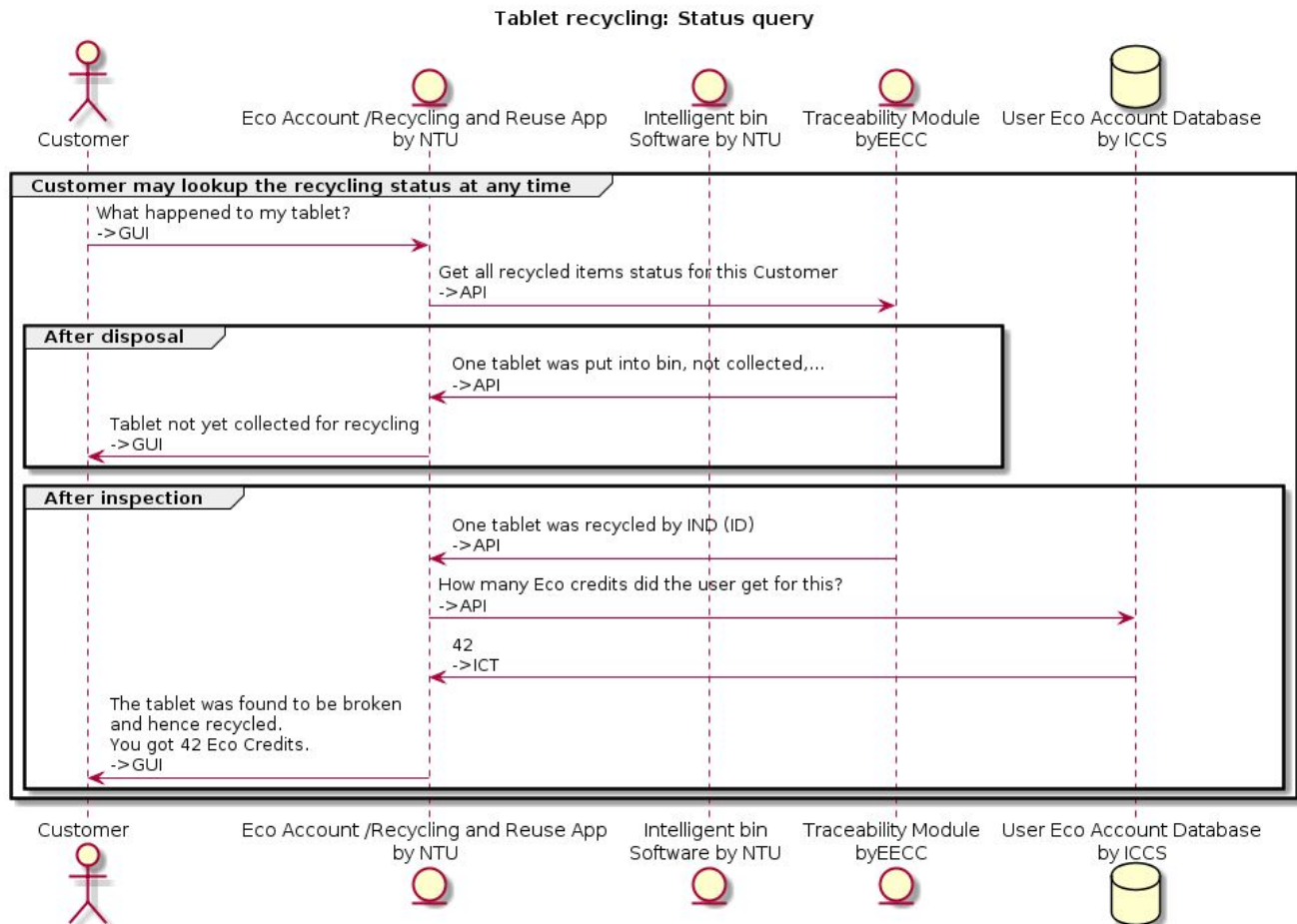


Figure 3: Sequence diagram for information query of electronic devices disposed at a smart bin.

After IND collects the electronics item from the bin, it is inspected and rated at IND's plant. The Traceability Module captures the data through a user interface and then pushes it to the CIRC4Life ICT platform where eco credits are calculated and stored (see [Figure 4](#)).

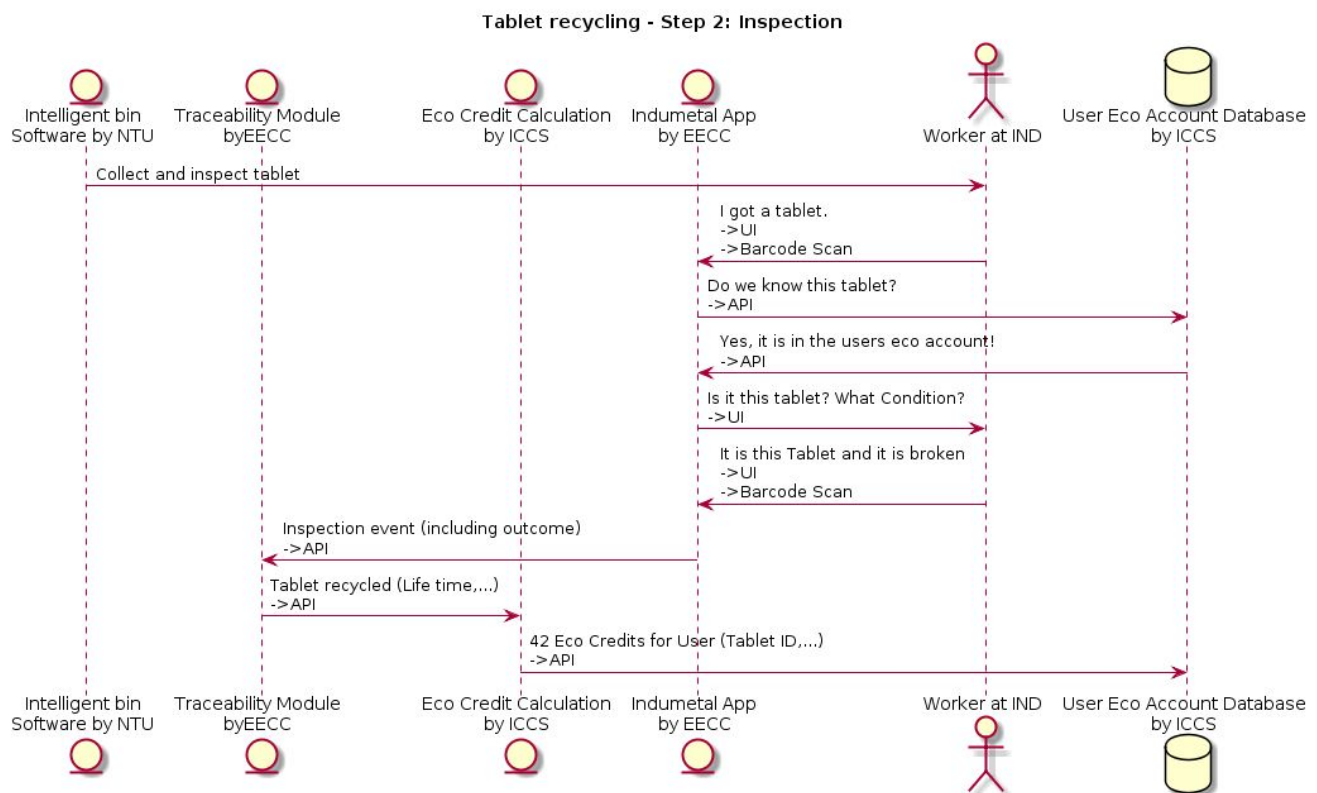


Figure 4: Sequence diagram for the inspection of electronic devices at IND's facilities.

Food Waste Recycling

The sequence of events and also the data flow for this use case are very similar to the one depicted in Section [Electronics Recycling and Reuse](#). Here the collection of bio waste is demonstrated and reuse/remanufacturing is not an option. Nevertheless, the material is disposed in a suitable bag, rated and eco credit is provided based on the recyclability of the waste. For details of this use case, please also consult Deliverable 2.2.

Solution Strategy

The goal of tracing ecological impacts with EPCIS technology as detailed in Section [Dynamic Eco Scores](#) poses a challenge. Since the EPCIS standard was not developed with the tracking of environmental impacts in mind, EECC has identified a big innovation potential in making use of the flexibility of the standard and developing an *EPCIS Extension for Ecological Impacts*. This was done in close cooperation with the LCA experts within the consortium, in particular CIRCE, Swerea and GS1G. The extension makes it possible to capture the essential information about resource usage, waste production, and other environmental or social impacts associated with any business event that is recorded via EPCIS. The extension is described in detail in Section [The EPCIS Extension for Ecological Impacts](#). It will be fully compatible to the upcoming version 2.0 of the EPCIS standard.

In order to demonstrate the functional potential of the extension, EECC implements it in all demonstrations that involve partners along the supply chain (T 6.2 / T 6.4 / T 6.5). Here traceability data will be collected, including data about ecological impacts for specific products within CIRC4Life Demos. This data is made accessible for the dynamic computation of eco scores. To this end, EECC deploys the Traceability Module as Software as a Service (SaaS) running on a German cloud managed by EECC. All Services are developed as



RESTful web services (RWS) and all user interfaces developed by EECC are web interfaces. This modern approach of developing software is most fit for the application in a quite dynamic context with requirements evolving along with the software as in CIRC4Life and with many different partners, processes and software systems that need to be connected and different user equipment such as mobile phones, tablets and Laptops that are used to display the interfaces. Web services and interfaces can be accessed regardless of operating systems, programming languages or other environmental parameters at the partner's or user's site.

The Traceability Module also uses a service oriented design internally. The components are separated into largely independent web services and groups of related services run in separate docker containers. This allows for a clear separation of the software components and provides maximal reusability of components also in other contexts. It also makes the system more resistant to failures of individual components.

The user interfaces are internationalised and localised for the regions where the demonstrations will be implemented.

System Scope and Context

The primary purpose of the Traceability Module is to capture and store data about specific events, i.e. what happens to specific items, along with metadata about the events, such as when and where the event happens. In addition, the Traceability Module collects as much information about ecological impacts as possible along with every event.

The scope of Traceability Module is limited to dynamic data, i.e. data which is not known a priori and which directly relates to an individual item or to a batch of items. An example for a traceability event is the production, processing, or shipping of a specific item at a specific point in space and time. Static master data, such as the volume or weight of an item, that is constant for all items of the same product class, is **not** in the scope of the Traceability Module. The EPCIS standard also covers such master data, but master data is imported from other sources, e.g. the GS1 Registry by the interoperability layer without going through the Traceability Module. The ecological data of primary interest in CIRC4Life is of the type of so called instance/lot data, i.e. data that becomes available at a certain stage along the supply chain and which does not change afterwards, such as the best before date. More precisely, impacts can be accumulated to get e.g. the total CO₂ released throughout the life cycle of a product or also across products in a certain production process. Since the EPCIS repository is, by construction, a sequential database that never deletes information, it is particularly suited for the aggregation of data of this type.

EECC develops user interfaces for capturing the data if this is needed to overcome the challenge of a low digitalisation level at the data source. Nevertheless, the role of the Traceability Module in the overall system architecture is that of a data provider, not of an end user tool. This means that the Traceability Module does **not** use the collected data itself to compute eco scores and much less does it provide a particular view on the data to the end user. Instead, EECC provides and connects to interfaces in the form of web service endpoints in order to share the collected data for the project partners to make good use of it. The primary partner for the Traceability module downstream the data flow is the CIRC4Life ICT Platform developed by ICCS, which is the central hub of information within the CIRC4Life project. From here data flows to the computing services and finally to the end user applications. A very rough sketch of the overall data flow is given in [Figure 5](#).

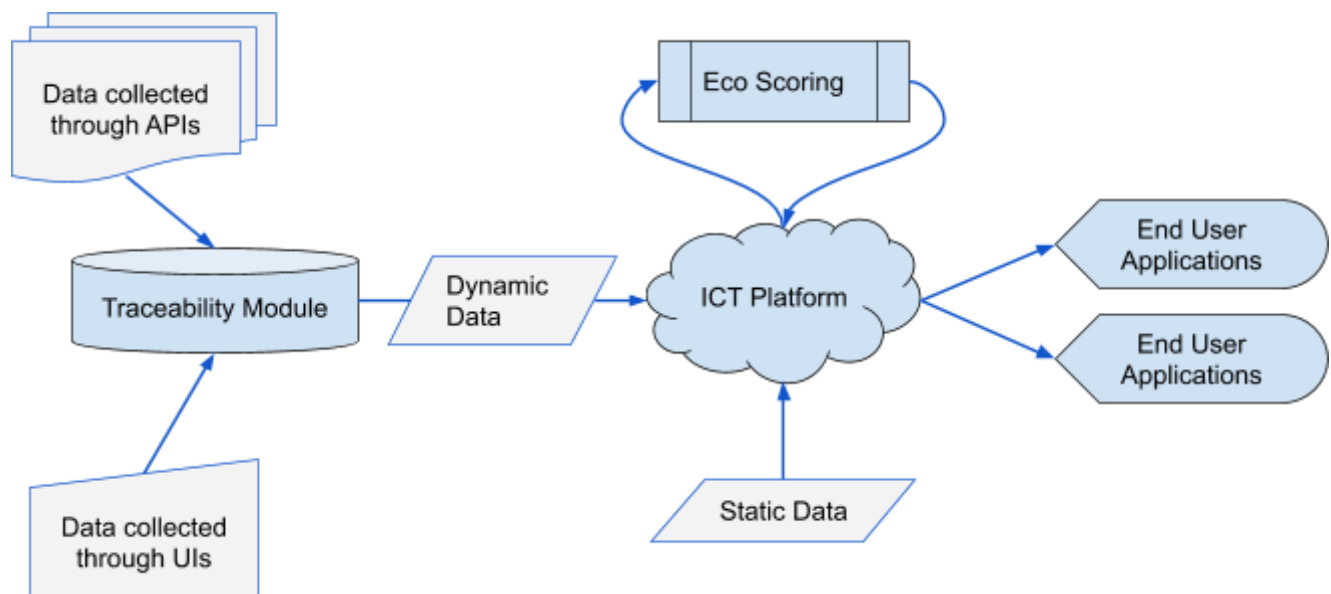


Figure 5: High level data flow diagram of the Traceability Module.

Dynamic Data Sources

The project partners within CIRC4Life cover all three sectors of the value circle (see Figure 6). The production of food and electrical equipment (ALIA, JS, KOS, ONA), the usage phase covered by the leasing/rental model developed by KOS and the reuse and recycling phase (IND, REC) are covered. See Figure 7 for a simplified pictogram of the value circle of e.g. consumer electronics, for which the recycling part will be demonstrated in T 6.3.

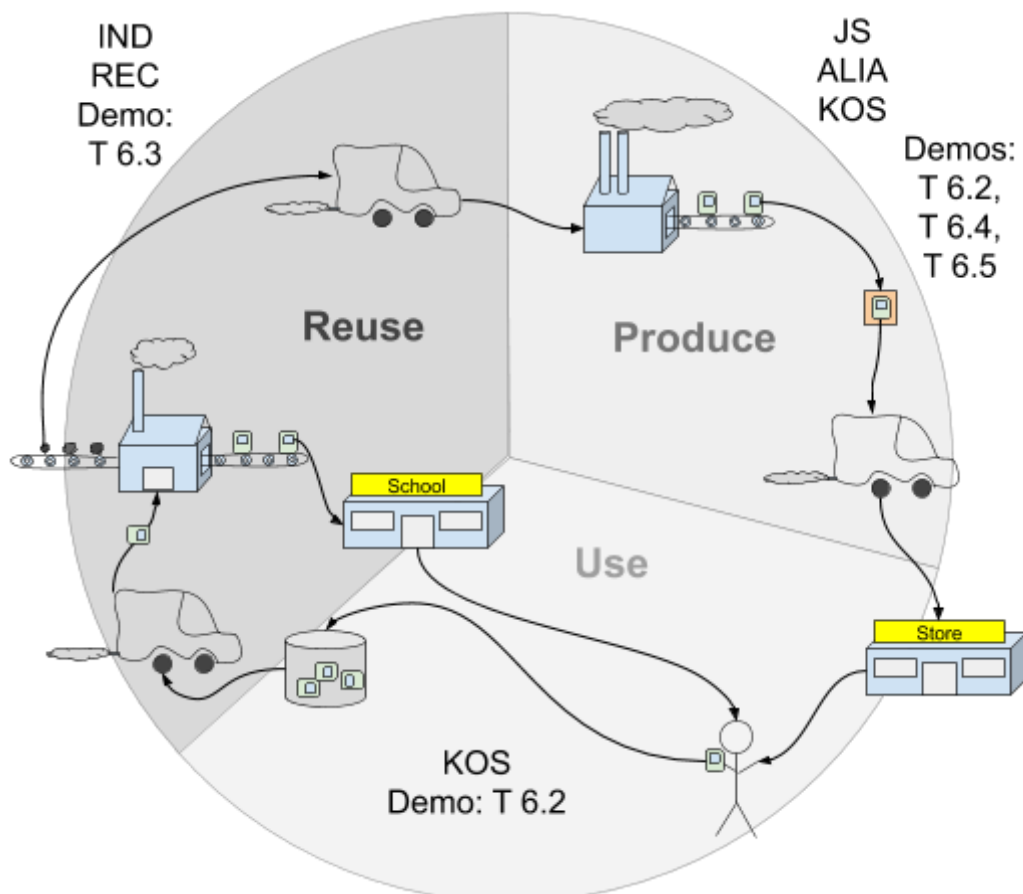


Figure 6: General value circle for CIRC4Life demos.

EECC has modelled all processes along these value cycles as traceability events using the EPCIS standard. Using this standardised description of processes, all partners can act as traceability data sources. Their data streams flow into the Traceability Module where the central EPCIS repository acts as a data lake. Drawing from this pool, the Traceability Module aggregates and transforms the information and brings it into formats that are easily and quickly accessible to all partners within CIRC4Life through web services. For example the latest known disposition and location of each tracked item is transformed and loaded into a caching database in order to provide quick access.

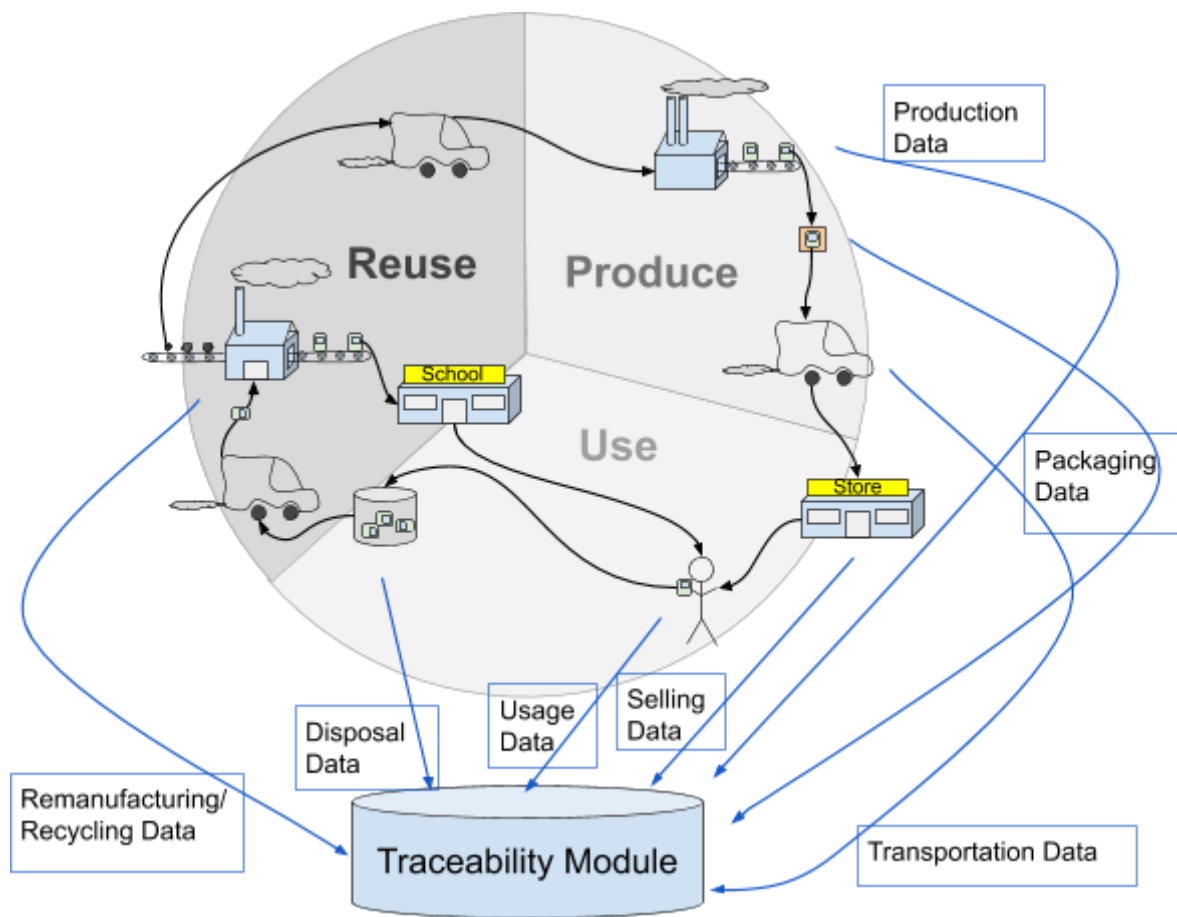


Figure 7: Simplified data flow for the electronics recycling and reuse demonstration to the Traceability Module.

Data Protection and Privacy

Personal (GDPR relevant) data must not be sent to the Traceability Module. Such data is not requested, nor used anywhere. It might be technically possible to erroneously input such data into free text fields, but doing so is explicitly forbidden. Hence the Traceability Module is not concerned with collecting or storing any kind of personal data and no special means of private data protection are implemented.

Nevertheless, all data is stored on servers at data centers in Germany and data access is only provided upon authorisation and only to CIRC4Life consortium members, unless there is a project management board decision to allow wider access. All members are requested to not leak any data to third parties under the same terms.

For considerations on how companies can effectively protect the data that is legitimately handled from unauthorized access, see Section [Ecological Transparency and Data Ownership Along the Supply Chain](#) for a more detailed description of the problem and Section [Data Access Model](#) for the solution developed for the Traceability Module.

Building Block View

Abstracting the above Figure 7 yields the data flow diagram in Figure 8.

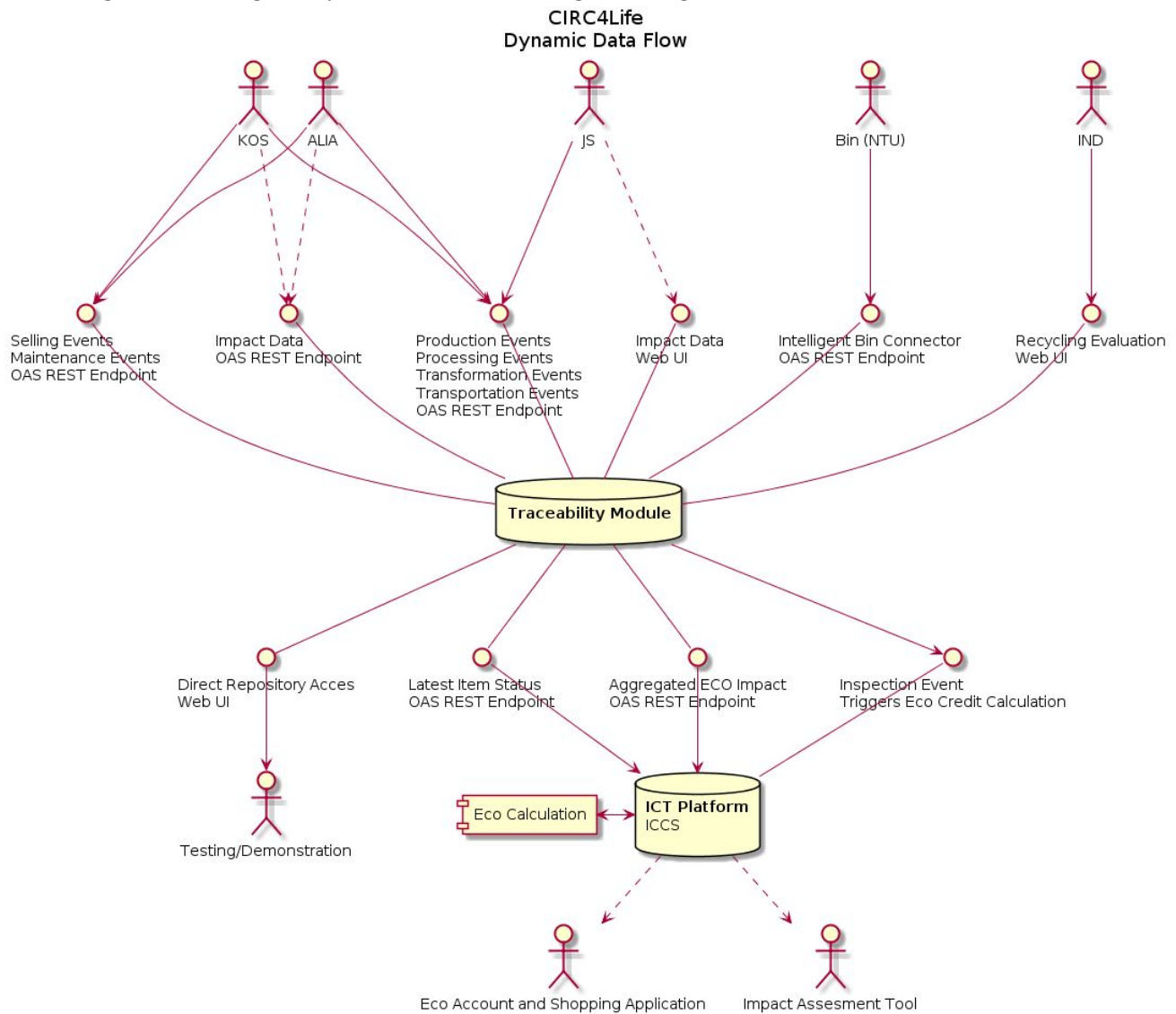


Figure 8: Data flow diagram through the Traceability Module.

Here the data flow has already been split into input into the Traceability Module (capturing) and data output (accessing) layers and further into use case specific endpoints. From this data flow and endpoints picture, EECC derived the overall system architecture for the Traceability Module given in [Figure 9](#).

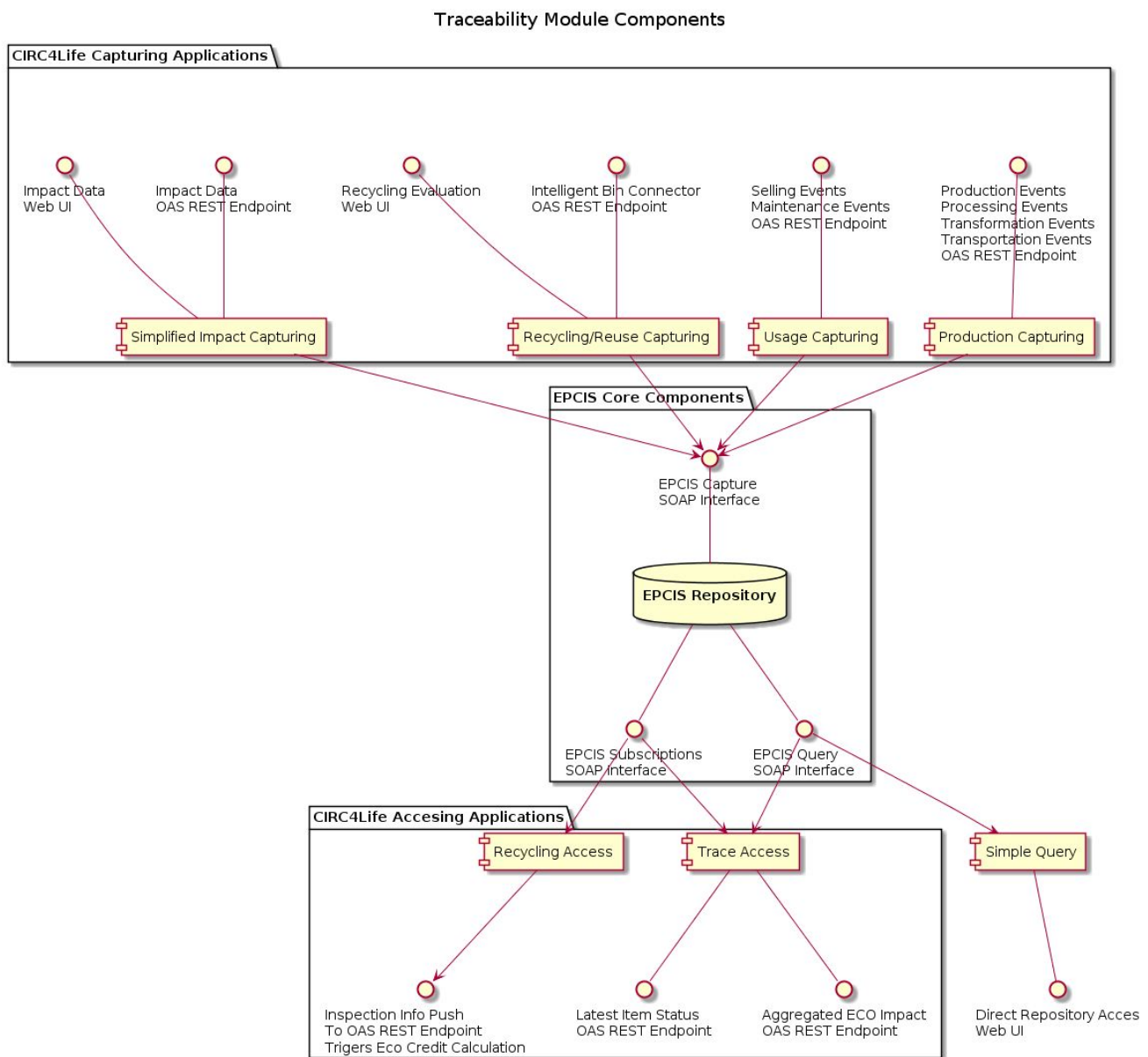


Figure 9: Main building blocks of the Traceability Module.

The individual components are described in detail in the following sections.

Design and Development of the Core Components of the Traceability Module

The EPCIS core system of CIRC4Life is able to provide the capture and query functionalities which are specified in the EPCIS Standard (GS1 Global, 2016) [3]. It fully supports the standard EPCIS data format and, in addition, the specific extensions EECC developed within CIRC4Life WP 5. This is in particular the ECO Extension is defined in Section [The EPCIS Extension for Ecological Impacts](#). The EPCIS core system has to store all events in a persistent way and to accept requests to the standard SOAP query interface and provide a response containing all matching events within near-real-time.

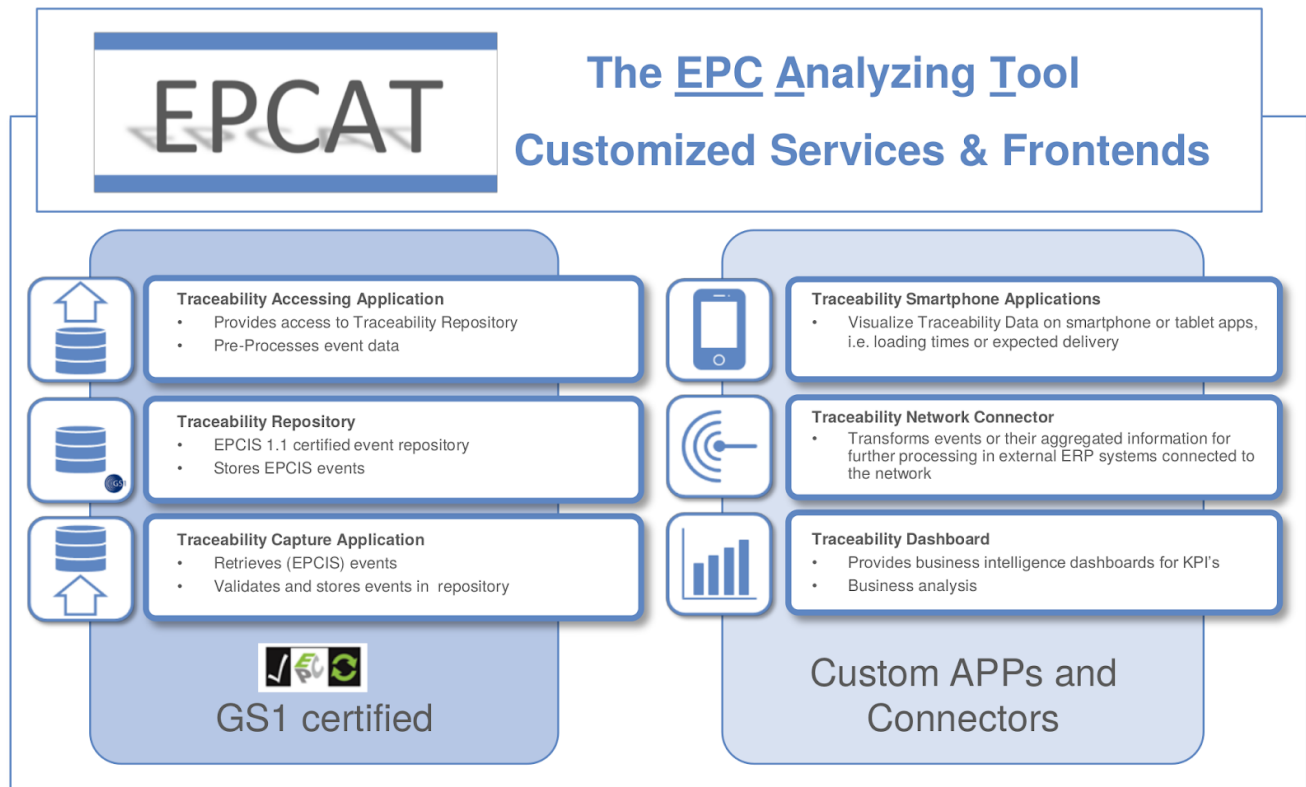


Figure 10: Components of the EPCIS core system of the Traceability Module

The EPCIS core system in CIRC4Life will be provided by EPCAT, a fully compliant EPCIS repository implementation by EECC (see [Figure 10](#)). EPCAT is provided by EECC as software as a service on servers at data centers running in Germany.

Design and Development of Web Services

In order to collect data, EECC has defined interfaces which enable supply chain partners to connect to the Traceability Module. EECC has developed all interfaces as RESTful Web services (RWS) utilising open API specifications (OAS, formerly known as Swagger) in order to make it as easy as possible to integrate with the traceability services. The service endpoints collect data in a narrow context and can hence afford to be rather simple and easy to use, compared to the generic standard interfaces. The capturing services then take care of adding the contextual information and transforming the data into standardised EPCIS events. EECC has also developed the access points as RWS and provides data pushing services wherever those fit into the general data flow model and the partners needs.

In addition to these more modern approach of using RWS and OAS, EPCAT also offers the full set of SOAP interfaces defined by the EPCIS standard. These interfaces can be used in a more generic way to capture, verify and validate any traceability events, or to query for any stored events. This way, the services stay flexible to use and extend beyond the concrete events and use cases envisaged in CIRC4Life. Furthermore, partners might choose to run their own EPCIS (core) repository in order to keep full control over their data as discussed in detail in Section [Data Access Model](#).



Capturing Services

As shown in Section [Building Block View](#), see in particular [Figure 9](#), EECC has grouped the capturing service endpoints by life time phase into three capturing applications. The applications target the production, use, and reuse/recycling phase, respectively. Additionally, EECC anticipates the need for simplified interfaces that enable the partners to enter dynamical data about ecological impacts that cannot be directly related to concrete business events. The latter would ideally not be used, but practical experience shows that it is not always possible to get the full event context for all data.


Since the applications are developed in close cooperation with the CEBMS with a particular focus on the planned demonstrations, the recycling application is by far the most advanced. This reflects the quite mature state of the recycling/reuse CE business model and the tablet recycling demonstration. For the other business models and demonstrations, WP 5 had to postpone some of their planned resources until the use cases and requirements become more clear as the work in the respective WPs progresses.


Simplified Impact Capturing


A simplified interface has been developed in order to enable the partners to record data about ecological impacts that is already accumulated and cannot be directly related to concrete events any more. Some supply chain partners might not trace every business event in detail and hence their data about the environmental impact of a product for might not fit into the ideal scenario, where each impact is directly associated with a business step. To cater for this case, EECC has developed a generic API to capture environmental input of an item without associating it to a specific business step. The documentation of this end point can be found in Section “Simplified Impact Capturing Endpoint” of [Appendix 2](#). This way of recording item specific impacts in a timely manner will still enable most of the benefits of [Dynamic Eco Scores](#) and is hence much preferable to not getting such accumulated data at all.

[Figure 11](#) shows a wireframe sketch of a user Interface for submitting data to this web service endpoint. At the current stage of the development, it seems that having the API will be enough for the purposes of the demonstrations within CIRC4Life and hence the implementation of the user interface is currently not planned. As mentioned above, depending on further development of the business models and demonstrations, EECC has set some capacities aside which will be used to concretely implement the UI if this turns out to be beneficial for a concrete case.

Trace Environmental Impact

 2019-05-05

 13:30:55

 urn:epc:id:sgln:4047111.12345.0001

Item urn:epc:id:sgtin:4047111.012345.012345678901

Resources
 Electricity 10 WH
 Water 1 LTR

Waste
 Lead 0.1 GRM

Transportation
 1000 KMT truck

Add Resource Consumed

[Add](#) [Cancel](#)

Add Waste Produced

[Add](#) [Cancel](#)

Add Transportation

[Add](#) [Cancel](#)

Submit

Figure 11: Generic Impact Capturing Web UI

Usage and Production Capturing APIs

The APIs developed for the production and usage phase follow a straightforward pattern. EECC has developed a specific endpoint for each type of event to which the partners can POST the event data, including impacts. More specific connectors to ALIA's systems, which are already capturing traceability information, are currently discussed and will be developed shortly, as soon as the details will have been negotiated. The "Object Event Capturing Endpoint" (see [Appendix 2](#)) can in principle handle event data in a very generic way, but it will likely be necessary to develop endpoints more adapted to ALIA's systems.

For the case of capturing usage data from lamp rental/leasing that is currently developed by KOS, it is still being under discussion what is to be tracked exactly. KOS is currently in the process of designing the business model and also the lamps, hence it is not yet clear what kind of sensors will be used and which data will be gathered and how. Once the case is more clear, EECC will develop connection endpoints to KOS's systems in order to retrieve the traceability data, in particular the eco impacts.

Farming UI

EECC had some very productive discussions with JS and also input from CIRCE about the data collected for their social LCA study. Based on these discussions, EECC has developed the concept for a web UI through which small farm businesses with a low digitalisation level can digitalise and hence trace their processes manually. In CIRC4Life, this UI will be used to compute dynamic ecological scorings for JS. See Section [Vegetables Production](#) for the details of the user story and [Figure 12](#) for a wireframe of the UI. The "Object



Event Capturing Endpoint” (see [Appendix 2](#)) can handle the data from the UI, although a more specific endpoint is under consideration. The idea is, that the list of impacts can be extended by clicking the plus icon and using a dialog similar to [Figure 11](#).

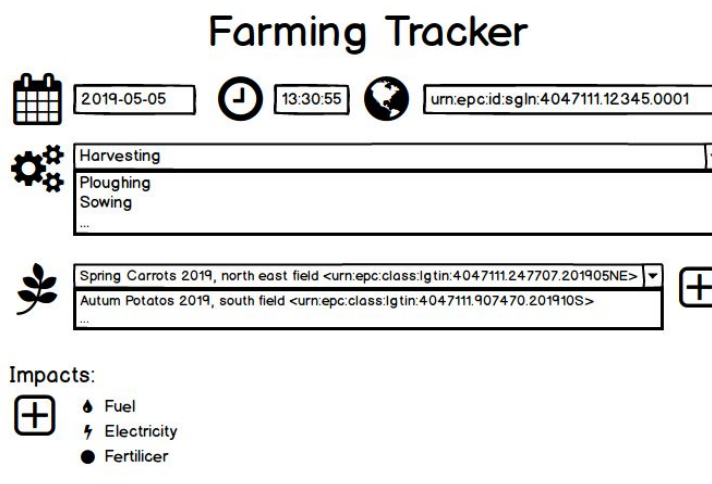


Figure 12: Farming Web User Interface Wireframe

Recycling and Reuse Capturing Application

As mentioned above, the recycling and reuse capturing application is way more advanced in development than the other capturing applications, due to the underlying business model and demonstration case being much more clearly developed by the partners for the time being. The purpose of the application is to keep track of all events that happen to an item starting with the disposal by an end user.

In the CIRC4Life demonstrations, disposal happens into an intelligent bin, which facilitates the tracking. EECC has developed an endpoint specifically to accept the data from the bin's control module as programmed by NTU. See [Appendix 2](#) for the documentation of this “Bin Disposal Event Receiving Endpoint”. See [Figure 2](#) for the sequence diagram of the user story for tablets. The more general story also includes bio waste recycling. As discussed with ALIA, the demonstration owners of this case, the steps in the sequence are essentially the same.

The next step is the collection and inspection/rating of the waste. For the tablet case, see [Figure 4](#). Again, the procedure for bio waste is planned to be similar, only the data input into the Traceability Module might happen by other means, i.e. by an to be defined API/connector or a more simple user interface than in the tablet case.

In order to get the inspection data from IND, EECC has developed a web based user interface. See Section [Recycling Evaluation Capturing Web Interface](#) below for the details.

The UI is connected via a service endpoint to the Recycling/Reuse Application which transforms the data into the EPCIS standard format and delivers into the EPCIS core via the standard SOAP interface. All components are running as SaaS in order to deliver the traceability data to the EPCIS core. This is the general setup used for all capturing endpoints.

Once the data arrives at the core, a subscription forwards it to the accessing application where it is transformed and pushed to the ICT platform. To this end, ICCS and EECC have developed suitable endpoint to integrate the Traceability Module with the CIRC4Life ICT Platform (T5.4). The overall development for the



various connection points is ongoing, but the endpoint to send the recycling evaluation data from the Traceability Module to is already well developed and ready to use. The full data flow from the evaluation UI through the capturing application, EPCIS core, and accessing application to the ICT platform using the commonly developed web service was implemented up front to serve as an example and system test case for similar data flows being implemented.

Recycling Evaluation Capturing Web Interface

It was discussed how traceability (WP 5) can be put to good use for the CEBM Collaborative Recycling/Reuse (WP 2) at the CIRC4Life innovation camp in Krakow. Since the digitalisation level at the recycling facilities is not very high, it was found that a user interface needs to be developed in order to be able to capture the relevant data for the demonstration (T 6.3) of this CEBM. Consequently, EECC has developed a simple web UI through which the workers at the recycling facility can capture the data from the barcode printed by the intelligent bin and add information about the end of life state and estimated lifetime of the product. A handheld (USB) barcode scanner can be used with the web UI. See [Figure 13](#) below for screenshots. The UI is deployed and functional. Please have a look at <https://circ4life.eecc.info/> for a live demonstration.

The screenshot shows a web browser window with the URL <https://circ4life.eecc.info>. The page features a circular logo with the text "CIRC4Life" and icons representing various electronic products. Below the logo, there is a form for data entry:

- BinBarcode**: A text input field.
- State**: Radio buttons for "No EEE item" (selected), "Working", "Damaged", and "Broken".
- Estimated lifetime**: A text input field with a spinner and a "Years" unit button.
- EEE Category**: A dropdown menu with the text "Please pick an EEE".
- Buttons**: Two blue buttons labeled "GTIN" and "Manufacturer & Model".
- GTIN**: A text input field.
- Serial Number**: A text input field.
- Submit**: A blue button.

Figure 13: GUI for Inspection of disposed EEE items at IND.



Accessing Services

Dynamic Eco Data

The most important service endpoint developed is the one by which all information about the ecological impacts of an item is provided. Making this data available within CIRC4Life is the primary goal of the Traceability Module, since this enables the dynamic (online) calculation of eco scores (eco points, credits, and debit as well as possibly other scores) for each individual item that is traced throughout the project. The “Total Impact Endpoints” (see [Appendix 2](#)) have been developed to deliver this data. However, the exact data flow and whether the endpoint as developed by EECC is in this form useful to NTU could not yet be discussed.

Item Status

EECC has implemented a service endpoint through which it is possible to query for the status, i.e. the last known disposition and location, of an item. This can be used in the demos to implement simple tracking user stories as for example described in Section [Electronics Recycling and Reuse](#), [Figure 3](#). Consult the “Item Status Endpoint” sections in [Appendix 2](#) for the documentation of the flavours of this endpoint.

Bin Filling Overview

EECC have developed an API through which the filling levels of all bins is made available. The purpose of this web service is to support the recycler in planning the collection tours. EECC propose a simple map-like user interface (see [Figure 14](#)) order to support this user story. The details are still to be discussed with the demonstration owners.

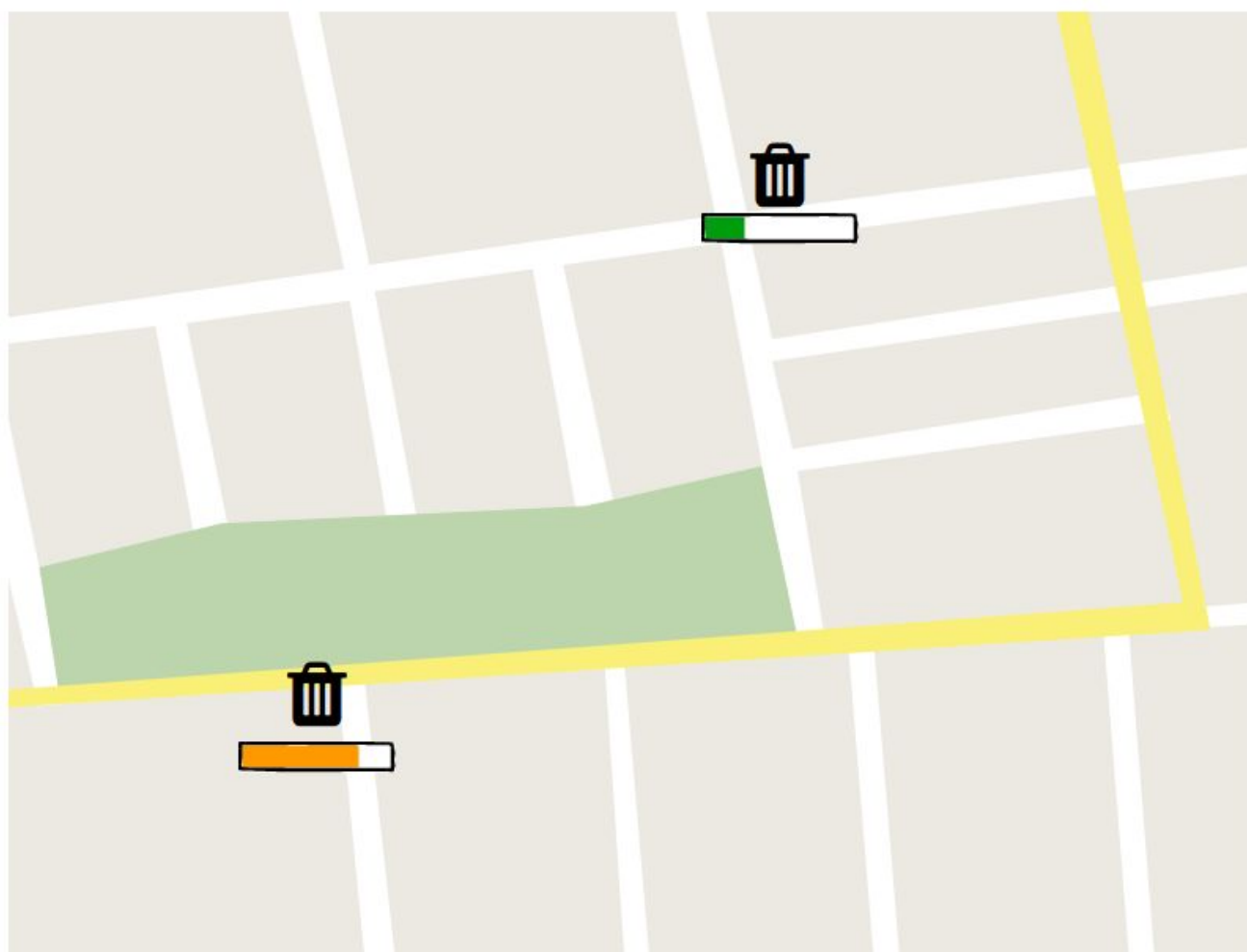


Figure 14: Map like UI for collection tour planning

Design and Development of Core Features of the Traceability Module

Of course, EECC has developed the inter-connection between the Traceability Module's internal components as shown in Section [Building Block View](#). Here the standard EPCIS SOAP interfaces are used and in particular a subscription which facilitates the continuous data flow from the EPCIS core (EPCAT) to those accessing applications which use caching or are pushing data to the ICT platform. Internal APIs to connect the web user interfaces to their services have been part of the UI development. This is state of the art technology and will not be described in more detail here.

Instead, the focus of this section is on novel traceability innovations developed specifically in this project. In particular, EECC has developed a new concept to enable all partners to maintain full control over their data and fine granular access rights management while using standard EPCIS tools. The details are described in Section [Data Access Model](#) below. This concept might be needed in order to enable adaption of the idea of sharing dynamic ecological data beyond a relatively small consortium as in the CIRC4Life demonstrations. Notice that it is applicable to EPCIS based traceability solutions in general and addresses a problem often occurring in practise.



The [EPCIS Extension for Ecological Impacts](#) was developed specifically to collect and share data about ecological impacts. As explained in section [Dynamic Eco Scores](#), this collection of primary data is essential for the computation of dynamic eco scores, which is a major innovation of the CIRC4Life project as a whole.

Data Access Model

A problem similar to the challenge identified in section [Ecological Transparency and Data Ownership Along the Supply Chain](#) has been discussed in [\[1\]](#) and [Appendix 1](#). How can the sharing of (the relevant parts of) traceability data along the chain of custody be enabled while, at the same time, protecting the data from unauthorised access and preventing the leakage of business relevant information, such as production volumes? Part of this problem is also tackled in the ongoing development of the Version 2.0 of the EPCIS standard. After private discussion with Ralph Tröger and Matthias Guenther, authors of the aforementioned papers, EECC has developed an EPCIS 1.2 compatible way of using the central ideas from [\[1\]](#) without the need to create a network of services on top of the EPCIS repositories that hold the actual data. Like in [Appendix 1](#), the approach proposed in this section is purely peer to peer, but using only established standards, namely EPCIS 1.2 [\[3\]](#) and named identifiers [\[2\]](#).

CIRC4Life uses a centralised ICT architecture. All traceability data is gathered in a central EPCIS repository at the heart of the Traceability Module. In this setup, state of the art technology can be used to provide fine grained access control using an appropriate user/group/role models by providing single sign on and token based access, as mentioned in Section [Data Protection and Privacy](#). A free software implementation of the Kerberos protocol will be used for this purpose and the Traceability Module implements a suitable client to consume the access tokens issued by the master service at the ICT platform.

Nevertheless, it is foreseeable that adaption of the traceability solutions beyond CIRC4Life might make a more decentralised approach necessary, where at least some of the companies along the value chain host their own EPCIS repositories in order to keep full control over the stored data. EECC has therefore adapted some of the ideas from [\[1\]](#) and [Appendix 1](#) to develop a solution for this setup.

It is assumed throughout this section that the user who wants to gather information about ecological impact (or some other traceability information for that matter) has an item identifier and knows a URL to an endpoint of a host that holds some traceability information. For example, a customer might have bought an item with an SGTIN from a vendor who offers his traceability information. The entry point might also be a named identifier (see below). It is further assumed that the chain of EPCIS events recorded for this item is connected, i.e. starting from the initial event, each company hosting traceability information about this item or its compounds holds references to the companies up or down the chain of custody such that all involved partners can (indirectly) be reached.

The system of lookups described in the following is envisaged to be implemented in the client software. Consequently, it will be transparent to the end user, such that the user experience is the same whether the data comes from a single source, as in CIRC4Life, or whether the protocol described in this section is executed in order to gather the data.

Named Information AKA Hash URIs

An important technology that can be used in order to conceal data content while at the same time providing a proof of integrity is the URI scheme to name things with hashes as described in the IETF's RFC 6920 [\[2\]](#). A brief summary of the main technical points is given here for the reader's convenience.

The URI scheme identifier is "ni" which stands for named information. An authority (such as a domain name) may optionally be given in order to retrieve the referenced resource. For the scheme developed here, the authority shall always be included. It is followed by the specification of the hashing algorithm (typically



“sha-256”) followed by the hash value of the resource. In the present case, the data to be hashed will always be a string and usually a URI, but it may also be a fixed string representation of a complex XML element. To this hash value, a query string may be appended. In summary, the named information (NI) URI format is

`ni://authority/alg;val?query-string`

which by definition ([RFC 6920](#)) refers to a lookup at

`https://authority/.well-known/ni/alg/val?query-string`

which might return a redirect (HTTP status code 3xx) to the actual URL of the resource.

For example¹

`ni://circ4life.eecc.info/sha256;yIY5qGjUN52l48ROiNJgvib79gcKVldwE8wXAwYkd-Y?ct=text/plain`

can be looked up at

<https://circ4life.eecc.info/.well-known/ni/sha256/yIY5qGjUN52l48ROiNJgvib79gcKVldwE8wXAwYkd-Y?ct=text/plain>

to yield

`urn:epc:id:sgtin:4047111.012345.1234567890`

The hash of this SGTIN URN string is indeed `yIY5qGjUN52l48ROiNJgvib79gcKVldwE8wXAwYkd-Y`, hence the receiving party can be sure that the correct information has been received.

Security Gained by Using Named Information

A cryptographic hashing function, such as sha256, has the property of concealing the input. No information at all about the input can be deduced from the output. Being a function, the hash still produces a unique output whenever fed the same input, hence it can be used to check data integrity. This leads to the well known scheme of publicly disclosing the hash of some piece of sensitive data while keeping the data itself secret. Upon authorised request, the data can be handed out and the requesting party can check the integrity by computing the hash and comparing to what has been published.

The additional advantage of using named information URIs, in particular including a host name, is that it canonically translates into a URL from which the referenced data can be requested. The owner of the host can guard the data by implementing identification/authorisation requirements and access management as he sees fit.

Thread Model

If the amount of data to be concealed is very small, a typical attack by an unauthorised party interested in the data is to just brute force hashing all possible values until the outcome matches the published one. When hashing static IDs, such as GLNs, there is the additional problem that the same hash for the same ID will appear quite often.

For example, if a company wants to conceal the GLN of a trading partner by using a NI, an attacker will likely be able to narrow down the list of possible values to a very small set by just looking up the GLNs of plausible partners. Hashing all candidates will unconceal the GLN in no time.

¹ All example hashes in this document are generated using the standard linux sha256sum implementation <https://linux.die.net/man/1/sha256sum> and converted to base64URL (from the standard hex output) via `xxd -r -p` <https://linux.die.net/man/1/xxd> and <https://linux.die.net/man/1/base64> (changed to URL form, see <https://www.rfc-editor.org/rfc/rfc4648#page-7>).



As explained in <https://www.rfc-editor.org/rfc/rfc6920.html#section-10>, the named information scheme was developed to provide a proof of integrity and matters of effectively hiding or securing information are out of scope.

The well known solution to this problem is to use salts. Salting a hash means that a random nonce is appended to the data to be hashed before the hashing. The effect is similar to encryption. Without knowledge of the salt, it is impossible to produce the hash value even if the original value is known or correctly guessed, provided the salt is long enough (has enough entropy) and unknown to the attacker.

See section “[Double Envelope](#)” for the concrete implementation in this context.

Solution

After describing the general ideas, this section describes the setup which solves the mentioned problems.

In order to run a distributed Traceability Module, each company may run its own EPCIS Core as defined in section [Building Block View](#). Additionally, lookups as defined in section [Named Information \(Hash\) URI](#) must be supported, i.e. requests to `https://host/.well-known/ni/sha256/{hash_value}` to a suitable company owned host name need to be answered. Answers to all queries, i.e. named information as well as standard EPCIS queries, are subject to identification and authorisation of the requesting client. This way, the companies running their own EPCIS core retain full control over who may access their data.

The data owner may, at his own discretion, conceal any value of any XML element in EPCIS events stored in their repository by replacing the concealed XML data with a named information (NI) URI. This way of sanitization can for example replace the GLN or some or all of the EPCs, if the company does not want to reveal which asset or which location is involved.

Companies are free to implement any authorisation mechanism they see fit and protect more sensitive data inside EPCIS events with stronger authentication requirements. For example, the concealed EPCIS events might be public, while the included NIs are only be resolved upon authorized request.

Double Envelope

An important point is that the authority component of the NI should always point to the data owner's own host. This allows to reveal only less sensitive information and then step by step more sensitive data. For example, the ecological impacts, most relevant to CIRC4Life, can be shared with a broad audience while keeping more sensitive data subject to stricter authorisation constraints. Applying more and more strict rules in each successive round of deeper NI lookups, arbitrary access models can be implemented.

One place where references to partners up or down the supply chain may occur are host names in the NI URN. For example, if the input EPCs into a transformation event are concealed in a sanitized event, one should not refer to foreign hosts directly, even if that is the correct source to query for the EPC. If a foreign host name is revealed, the business relation to that partner is leaked. Instead, the NI URN should be concealed as the information content of another NI at the companies own host. For example, instead of including

```
<epc>ni://myPartnersHostName/sha256;Yn-49Dnss3FRxJQrpmmDZ5fwYyNiJjMXHeL1VNpuLJI</epc>
```

in an EPCIS event, one should use

```
<epc>ni://myOwnHostName/sha256;EXnMeueUZi9lCeDVxu8LPxxbwnHoMYb8G-oHUC28C_U</epc>
```

A lookup to the URL associated with the latter, i.e.

```
https://myOwnHostName/.well-known/ni/sha256/EXnMeueUZi9lCeDVxu8LPxxbwnHoMYb8G-oHUC28C_U
```

then yields the above NI referring to the partner. Importantly, this final lookup can again be guarded. Only after checking that the requesting party is allowed to access this business relevant piece of information is the



NI URN resolved, otherwise the host returns a 401/403 HTTP error code indicating that the request is not authorised.

As mentioned in section "[Thread Model](#)", a particularly interesting case is that of referring to another company directly by e.g. using their GLN as the value in one of the EPCIS event fields. For example, the source or destination fields in shipping/receiving events or business transactions might contain GLNs of partners, which is at the same time quite sensitive information and one which is vulnerable to brute force hash reversal attacks.

Here, the same double-lookup idea can be used in order to salt hashes with a query parameter. This avoids the problem of hashing short/static information such as GLNs mentioned in Section [Thread Model](#) and at the same time uses the original NI scheme exactly as defined in [\[2\]](#). More concretely, if the SGLN

urn:epc:id:sgln:4047111.12345.0001

which has a sha256 hash of JD80Y6vKGJ1D1H6H5gdQF8MZr_EEWB02gBEo8EPCuDE is to be concealed, one should not use

ni://myOwnHostName/sha256;JD80Y6vKGJ1D1H6H5gdQF8MZr_EEWB02gBEo8EPCuDE

directly, as this hash is vulnerable to brute force reversal. Instead using

ni://myOwnHostName/sha256;AWNwq9Cqb-biQ19hC670i-pHHAD1Qpq32bE39xrcaHI

which is (upon authorised request) resolved to

ni://myOwnHostName/sha256;JD80Y6vKGJ1D1H6H5gdQF8MZr_EEWB02gBEo8EPCuDE?salt=634202135

means using a salted hash. A random number has been added as a "salt" query parameter. This salt effectively defeats the brute force attack and is just ignored by the host when answering the corresponding URL query.

Comparison to the State of the Art

The process of concealing data within sanitized EPCIS events by using named identifiers as developed by the EECC within CIRC4Life and described in this section can be viewed as a simplified version of similar ideas presented in [\[1\]](#).

In contrast to that approach, no extra visibility and discovery services are needed on top of the core EPCIS repositories maintained by each data owning company. As in [Appendix 1](#), a pure peer to peer approach is used, but the scheme proposed here uses EPCIS for the "node" as well as the "data" such that no new formats have to be introduced and existing standards are utilized.

In the approach introduced in this section, the data owners keep full control but also full responsibility not only over who may access their data but they are also responsible for sanitizing events and concealing the parts they find relevant. These considerations have to be included in the design and setup of their local Traceability Modules. The nested data storage model, which is more complex in the current setup than in [\[1\]](#), needs to be baked into the capturing applications. Another drawback is the more complex lookup. In this model, a client has to identify himself to each company and retrieve the EPCIS events in multiple steps and on top successively lookup the named identifiers.

EECC does not consider these points as major drawbacks. In general, the lookup process of the client is easily automated. For this use case, high speed is of less concern when looking up complete event chains across multiple companies. In CIRC4Life, the authorized client is e.g. the eco score calculating module, which only needs to look up the total dynamic impact data for a given item if new impacts have been recorded and a new calculation is necessary. Otherwise, cached eco scores can be used if high speed lookups are needed.



Including salts or concealing foreign hosts by using [Double Envelopes](#) makes it possible to store everything in a consistent way without the need of maintaining additional masking code tables as in [\[1\]](#), which offsets the more complex structure of the actual data storage.

Notice that the solution presented here does not require any extension to the existing EPCIS 1.2 standard [\[3\]](#). Also named identifiers are used exactly as specified in [\[2\]](#), with the slight modification of allowing for a salt query parameter, which is to be ignored in queries anyway. This means that the solution developed here is fully consistent with existing standards and uses only one extra feature (named information URIs) on top of the regular EPCIS core repository in order to achieve the goal of enabling every data provider to maintain full control over the data access and at the same time providing a usable way of gathering data from the whole supply chain.

Examples

Receiving an item

In the following example EPCIS event, the values of an EPC and a bizLocation id where replaced with double-enveloped named identifiers as explained in Section [Double Envelope](#). This way, the EPCIS event can be published without disclosing the actual epc of the item, the trading partner, or the location where it arrived. What is published is the information about the ecological impact, which is hence publicly associated to the NI (EPC). To get more information from the other supply chain partners, a client will have to identify himself to `circ4life.eecc.info` where the information owner can decide to grant or deny access to the EPC behind the named identifier. If access is granted, the foreign host and original hash are revealed and the client can go on querying the foreign host. The supply chain partner can now decide himself whether to reveal the information or not. This example illustrates that all involved parties keep full control over sensitive information such as their trade relations and the items produced in their facilities.

```
<?xml version="1.0" ?>
<epcis:ObjectEvent xmlns:epcis="urn:epcglobal:epcis:xsd:1"
xmlns:epcglobal="urn:epcglobal:xsd:1" xmlns:c4l="https://circ4life.eecc.info/epcis">
  <eventTime>2018-06-19T13:48:41.687Z</eventTime>
  <eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
  <epcList>
    <epc>
ni://circ4life.eecc.info/sha256;EXnMeueUZi9lCeDVxu8LPxxbwnHoMYb8G-oHUC28C_U
    </epc>
  </epcList>
  <action>OBSERVE</action>
  <bizStep>urn:epcglobal:cbv:bizstep:arriving</bizStep>
  <bizLocation>
    <id>
ni://circ4life.eecc.info/sha256;KQdhgBpHGSA1JlIOCTKzw50Bfp8o_FGi66kNkn1q3mSk
    </id>
  </bizLocation>
  <c4l:transportList>
    <c4l:transportation>
      <c4l:vehicle>http://purl.org/vso/ns#Truck</c4l:vehicle>
      <c4l:distance>
        <c4l:quantity>500</c4l:quantity>
      </c4l:distance>
    </c4l:transportation>
  </c4l:transportList>
</epcis:ObjectEvent>
```



```
<c41:uom>KMT</c41:uom>
</c41:distance>
</c41:transportation>
</c41:transportList>
</epcis:ObjectEvent>
```

Hiding More Complex Information

If more complex information is to be concealed, one can also replace the value of complex XML elements with named identifiers.

For example, the whole EPC list in Section [Bin Disposal](#) could be replaced by

```
<epcList>
ni://circ4life.eecc.info/sha256;1f1K68UnrPSCqL_Rfg3ZRTRET-_yBvZRtEabAboiK1A
</epcList>
```

where the named identifier encodes

```
<epc>https://circ4life.eecc.info/recycling/bin/obj/1234567890119051400001112345</epc>
<epc>urn:epc:id:sgtin:4047111.012345.12345678901</epc>
```

This simple example also reveals a subtle problem of hashing complex values. The exact formatting matters for hashing. Changing whitespace such as adding a space or using different line endings does not change the content, i.e. the data represented by the XML, but it changes the hash value.

This is a typical problem when hashing XML (see e.g. [\[7\]](#) for a discussion of the problem in the context of signing XML and [\[8\]](#) for a generic solution). In this use case, this is not of concern. The information owner hashes the content and provides it in exactly the form that yields the revealed hash. This way, a client requesting the information can check its integrity directly upon receipt.

The EPCIS Extension for Ecological Impacts

The CIRC4Life Eco Extension, which was first specified in D 5.1, Section 7, has been further developed. This extension to the EPCIS 1.2 standard [\[3\]](#) is the most important development of the Traceability Module for CIRC4Life since it enables to capture ecological impacts together with any business event recorded. Hence the data for accessing the life cycle of individual items can be gathered and processed. Another use case is to break down the ecological impact of a supply chain or a company into specific process steps in order to identify the hot spots in order to improve the overall impact most efficiently.

Making use of the extensibility of the EPCIS standard, all XML elements defined in the following schema can optionally be added to EPCIS events.

XML Schema Definition (XSD)

The following XSD formally specifies the extension to the EPCIS standard. Version 2.1.0 shown here is the latest version at the time of writing. The current version is available online at [\[10\]](#), where Semantic Versioning [\[9\]](#) is used to clearly indicate the type of changes to previous versions. An effort is made to ensure that the specification is not only machine readable, but at the same time human understandable by adding comments to explain all XML elements.



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:epcglobal="urn:epcglobal:xsd:1"
  xmlns:epcis="urn:epcglobal:epcis:xsd:1"
  xmlns:c4l="https://circ4life.eecc.info/epcis"
  targetNamespace="https://circ4life.eecc.info/epcis"
  elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation xml:lang="en">

      This document describes the eco-extension developed within the CIRC4Life research project.
      This project has received funding from the European Union's Horizon 2020 research and innovation
      programme under grant agreement No 776503.

      The purpose of this extension is to gather data about ecological impacts along with potentially
      any EPCIS event. This is done in order to enable a dynamic (online/real time) assessment of the
      ecological footprint and impact of an individual product based on primary data.

      <copyright>Copyright 2018-2019 European EPC Competence Center GmbH (EECC)</copyright>
      <licence>
        This document is licensed under a Creative Commons Attribution-ShareAlike 4.0 International
        License.
        See http://creativecommons.org/licenses/by-sa/4.0/ for details.
      </licence>

      <disclaimer>
        THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF
        MERCHANTABILITY, NON INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE
        ARISING OUT OF THIS SPECIFICATION. EECC disclaims all liability for any damages arising from
        use or misuse of this Standard, whether special, indirect, consequential, or compensatory
        damages, and including liability for infringement of any intellectual property rights,
        relating to use of information in or reliance upon this document.

        EECC retains the right to make changes to this document at any time, without notice. EECC
        makes no warranty for the use of this document and assumes no responsibility for any errors
        which may appear in the document, nor does it make a commitment to update the information
        contained herein.
      </disclaimer>
      <specification>
        EPCIS Extension for Ecological Impacts - Version 2.1.0
      </specification>
    </xs:documentation>
  </xs:annotation>

  <!--
  The following specifications from the EPCIS 1.2 standard may be fetched from
  https://www.gs1.org/docs/epc/epcis_1_2_schema-20160929/
  copies are kept at
  https://circ4life.eecc.info/doc/
  -->
  <xs:import namespace="urn:epcglobal:xsd:1"
    schemaLocation="https://www.gs1.org/docs/epc/epcis_1_2_schema-20160929/EPCglobal.xsd"/>
  <xs:import namespace="urn:epcglobal:epcis:xsd:1"
    schemaLocation="https://www.gs1.org/docs/epc/epcis_1_2_schema-20160929/EPCglobal-epcis-1_2.xsd"/>
```



```
<!-- simple types -->

<xs:simpleType name="lifeTime">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      (Estimated) time from manufacturing until disposal of an item or good
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:duration"/>
</xs:simpleType>

<xs:simpleType name="UNUKey">
  <xs:annotation>
    <xs:documentation>
      The UNU-Key describes the WEEE category of a device. See
      https://unu.edu/projects/e-waste-quantification.html#outputs
      Examples of relevance in CIRC4Life:
      0303 - Laptops (incl. tablets)
      0306 - Mobile Phones (incl. smartphones, pagers)
      0402 - Portable Audio and Video (f.i. MP3, e-readers, car navigation)
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>

<xs:simpleType name="HSCode">
  <xs:annotation>
    <xs:documentation>
      The Harmonized Commodity Description and Coding System, also known as
      the Harmonized System (HS) of tariff nomenclature is an internationally
      standardized system of names and numbers to classify traded products.
      UNUKeys correspond to (usually more than one) HS Code, but HS Codes
      are more fine grained and many HS codes do not have any corresponding
      UNU Key.
      See https://www.foreign-trade.com/reference/hscode.htm
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>

<!-- Resources -->

<xs:element name="resourceList">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      List of Resources consumed/used up in the business step
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <!-- Standard EPCIS -->
      <xs:element name="epc" type="epcglobal:EPC"/>
      <xs:element name="quantityElement" type="epcis:QuantityElementType"/>

      <!-- CIRC4Life specific -->
      <xs:element name="resourceElement" type="c4l:resourceElementType"/>
    </xs:choice>
  </xs:complexType>

```



```

</xs:element>

<xs:complexType name="resourceElementType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      ResourceElement is different from quantityElement in allowing for more general resource
      (water, electricity,...) than an EPCClass and additionally the source of each resource
      (renewable, primary/secondary, ...) may be specified.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="resource" type="c41:resourceType"/>
    <xs:element name="amount" type="c41:measure"/>
    <xs:element name="source" type="c41:sourceType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="measure">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      This type optionally contains a number with an optional unit of measure.
      It can be used to quantify countable or measurable quantities of resources
      (liters of water, Wh of energy, etc.) or waste, distances, etc.
      Utilising the flexibility in the UOM type, it is possible to also use any scale, also
      qualitative ones.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element name="quantity" type="xs:decimal"/>
    <xs:element name="uom" type="c41:UOMType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="UOMType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      This type is the same as epcis:UOM, but for the restriction to allow only certain units in
      epcis.
      It is strongly recommended to use a code from the list published in "Recommendation No. 20:
      CODES FOR UNITS OF MEASURE USED IN INTERNATIONAL TRADE" by the UNITED NATIONS ECONOMIC
      COMMISSION FOR EUROPE.
      If some other measure is used, the URI of the corresponding ontology SHOULD be used.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="resourceType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      Resources which do not fit into an EPCClass such as water, electricity, etc. can be specified
      in a resource element. It is usually preferable to use an EPC (class) for the used resource,
      if available.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

```



```

<xs:simpleType name="sourceType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      Where the resource is taken from, e.g. energy might be from a renewable or from a fossil
      source, materials may be primary or secondary (recycled), etc.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

<!-- Waste -->

<xs:element name="wasteList">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      List of waste produced in the business step
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <!-- Standard EPCIS -->
      <xs:element name="epc" type="epcglobal:EPC"/>
      <xs:element name="quantityElement" type="epcis:QuantityElementType"/>

      <!-- CIRC4Life specific -->
      <xs:element name="wasteElement" type="c4l:wasteElementType"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:complexType name="wasteElementType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      WasteElement is similar to ResourceElement but denoting waste produced instead of resources
      consumed
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="waste" type="c4l:wasteType"/>
    <xs:element name="amount" type="c4l:measure"/>
    <xs:element name="sink" type="c4l:sinkType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="wasteType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      Waste such as polluted water, exhaust fumes, etc.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

<xs:simpleType name="sinkType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      Where the waste is disposed (released, treatment, landfill, burning for energy,...)
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

```



```

        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

<!-- Transportation -->

<xs:element name="transportList">
    <xs:complexType>
        <xs:annotation>
            <xs:documentation xml:lang="en">
                If the environmental impact of transportation can not be broken down into resource
                consumption (fuel,...) and waste (exhaust fumes,...) then a transportList can be used
                instead.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="transportation" type="c41:transportElementType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:complexType name="transportElementType">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            How far and by which means
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="vehicle" type="c41:vehicleType"/>
        <xs:element name="distance" type="c41:measure"/>
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="vehicleType">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            The type of vehicle such as lorry, plain, boat, etc.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI"/>
</xs:simpleType>

<!-- Other Impact -->

<xs:element name="impactList">
    <xs:complexType>
        <xs:annotation>
            <xs:documentation xml:lang="en">
                The list of further impacts can be used to track social or ecological impacts that are not
                associated with resource consumption or waste production such as e.g. land usage.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="impact" type="c41:impactElementType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```



```

    </xs:complexType>
  </xs:element>

  <xs:complexType name="impactElementType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The type of impact together with an (optional) suitable measure.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="impactType" type="c41:impactType"/>
      <xs:element name="measure" type="c41:measure"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="impactType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The type of impact, such as land usage, social impacts, etc. Using an URI to refer to an
ontology within
        the semantic web.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI"/>
  </xs:simpleType>

</xs:schema>

```

Examples for Events in the Demonstration of the Recycling/Reuse CEBM

In this section, some examples for these EPCIS events that utilize [The EPCIS Extension for Ecological Impacts](#) on top of the EPCIS 1.2 standard [3] and its core business vocabulary [4] are given. The sequence of events exemplifies the process of recycling a tablet as developed in WP2 for demonstration in T 6.3. EECC has modelled similar event chains for all use cases for the demonstrations and business models currently developed in CIRC4Life, see D 5.1 for details.

Bin Disposal

After the Customer disposes his end of life electronics product into the smart bin, the bin sends the relevant information to the Traceability Module. To this end, EECC has generated a suitable web service (see Section [Recycling and Reuse Capturing Application](#) and “Bin Disposal Event Receiving Endpoint” in [Appendix 2](#)) that consumes the JSON structure as defined in “BinDisposalData” in [Appendix 2](#) and produces an EPCIS standard element like the following one (comments are added here for clarity, not actually generated).

```

<ObjectEvent>
  <!--
    This is an example of a bin disposal event. It records that a consumer places an end of life
    product electronic (e.g. a mobile phone) in an intelligent recycling bin. Apart from recording
    the actual data about the event, the eco extension is used in order to also track the energy

```




```

used by the bin.
-->
<eventTime>2019-05-24T10:00:00.0Z</eventTime>
<eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
<epcList>
  <!--
    The custom bin/obj namespace is used for the proprietary (i.e. non standard) barcode format that
    is unfortunately used by the intelligent bin operating in CIRC4Life.
  -->
  <epc>https://circ4life.eecc.info/recycling/bin/obj/1234567890119051400001112345</epc>

  <!--
    Example GTIN: (01)04047111123453 and Serial (21)12345678901.
    (Does not correspond to a real product.)
  -->
  <epc>urn:epc:id:sgtin:4047111.012345.12345678901</epc>
</epcList>
<action>OBSERVE</action>
<bizStep>urn:epcglobal:cbv:bizstep:arriving</bizStep>
<disposition>urn:epcglobal:cbv:disp:returned</disposition>
<readPoint>
  <!--
    The custom bin/obj namespace is used for demonstration purposes in CIRC4Life. In a real application an
    SGLN like urn:epc:id:sgln:4047111.12345.0001 should be used.
  -->
  <id>https://circ4life.eecc.info/recycling/loc/00001</id>
</readPoint>
<bizLocation>
  <id>https://circ4life.eecc.info/recycling/loc/00001</id>
</bizLocation>
<c4l:resourceList>
  <c4l:resourceElement>
    <!--
      This is an example for referring to an external ontology. It is recommend to use
      an existing ontology within the semantic web whenever possible.
    -->
    <c4l:resource>https://w3id.org/saref#Electricity</c4l:resource>
    <c4l:amount>
      <c4l:quantity>0.1</c4l:quantity>
      <!-- For the table of codes, see "Recommendation No. 20: CODES FOR UNITS OF MEASURE USED IN
      INTERNATIONAL TRADE" by the UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE-->
      <c4l:uom>WHR</c4l:uom>
    </c4l:amount>
    <c4l:source>
http://semanco02.hs-albsig.de/repository/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Not-Renewable\_Energy\_Source
    </c4l:source>
  </c4l:resourceElement>
</c4l:resourceList>
</ObjectEvent>

```

Bin Collection

The intelligent bin is able to indicate its filling level to the service endpoint, hence it is possible to provide an overview over the bins and their filling levels to the recycling company. Taking this into account, the recycler will collect the electronics.



```
<ObjectEvent>
<!--
This is an example of a bin collection event. It records that an item has been picked from
an intelligent recycling bin by the recycling company. The eco extension is used in order to track the
transportation of the item.
-->
<eventTime>2019-05-26T12:54:10.0Z</eventTime>
<eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
<epcList>
  <epc>https://circ4life.eecc.info/recycling/bin/obj/1234567890119051400001112345</epc>
</epcList>
<action>OBSERVE</action>
<bizStep>urn:epcglobal:cbv:bizstep:collecting</bizStep>
<disposition>urn:epcglobal:cbv:disp:returned</disposition>
<bizLocation>
  <id>https://circ4life.eecc.info/recycling/loc/00001</id>
</bizLocation>
<c4l:transportList>
  <c4l:transportation>
    <c4l:vehicle>http://purl.org/vso/ns#Truck</c4l:vehicle>
    <c4l:distance>
      <c4l:quantity>10</c4l:quantity>
      <c4l:uom>KMT</c4l:uom>
    </c4l:distance>
  </c4l:transportation>
</c4l:transportList>
</ObjectEvent>
```

Inspection Event

The most important event in the process is the evaluation of the item by the recycler. The information collected here has authority over claims by the user. For example, if a user claims that a tablet is working, but the recycler determines it to be broken, the latter is taken into account for the calculation of rewards for the user, such as eco credits.

```
<ObjectEvent>
<!--
This is an example of an inspection event. It records that an electronic item has been accessed by the
recycling company and records the state of the item using EPCIS standard core business vocabulary for
the disposition.
-->
<eventTime>2019-05-27T09:01:00.0Z</eventTime>
<eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
<epcList>
  <!-- Example for a custom namespace for a proprietary ID generated by the bin on disposal -->
  <epc>https://circ4life.eecc.info/recycling/bin/obj/1234567890119051400001112345</epc>
  <!-- Example for using a custom namespace to store information in a custom format, such as
       brand.model.serial, to track items for which the GTIN is not known
  -->
  <epc>https://circ4life.eecc.info/recycling/obj/Apple.iPhone_7.12345678901</epc>
</epcList>
<action>OBSERVE</action>
<bizStep>urn:epcglobal:cbv:bizstep:inspecting</bizStep>
<disposition>urn:epcglobal:cbv:disp:damaged</disposition>
<readPoint>
```



```
<!-- Example for a custom namespace for IDs identifying individual bins. In a real application,
      an SGLN should be used instead -->
<id>https://circ4life.eecc.info/recycling/loc/00001</id>
</readPoint>
<bizLocation>
  <id>https://circ4life.eecc.info/recycling/loc/00001</id>
</bizLocation>
<!-- Maximal possible life time. This estimate by the recycler is based on the manufacturing period of the
      model. It can be used to prevent the customer from cheating about the lifetime. -->
<c4l:lifeTime>P5Y</c4l:lifeTime>
<!-- The UNU-Key describes the WEEE category of a device. See
      https://unu.edu/projects/e-waste-quantification.html#outputs -->
<c4l:UNUKey>0306</c4l:UNUKey>
</ObjectEvent>
```

Repairing Event

If a returned (electronics) item is found to be “reusable”, i.e. not working/sellable as is but also not broken beyond repair, it can be refurbished to make it fit for a second life. This way of repairing and reusing an item is much more efficient than recycling and will hence be the preferred option. The repairing event is used to capture and store the information about this process step. In particular, information about the ecological impact can be captured as with all events.

```
<ObjectEvent>
  <!--
    This is an example of a repairing event. It records that an item has repaired/refurbished to give it
    a second life.
  -->
  <eventTime>2019-05-27T11:05:00.0Z</eventTime>
  <eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
  <epcList>
    <epc>https://circ4life.eecc.info/recycling/obj/Apple.iPhone_7.12345678901</epc>
  </epcList>
  <action>OBSERVE</action>
  <bizStep>urn:epcglobal:cbv:bizstep:repairing</bizStep>
  <disposition>urn:epcglobal:cbv:disp:returned</disposition>
  <readPoint>
    <id>urn:epc:id:sgln:4047111.123456.0001</id>
  </readPoint>
  <bizLocation>
    <id>urn:epc:id:sgln:4047111.123456.0001</id>
  </bizLocation>
  <c4l:resourceList>
    <!-- This is an example for using the eco extension to record energy consumption in a process -->
    <c4l:resourceElement>
      <c4l:resource>https://w3id.org/saref#Electricity</c4l:resource>
      <c4l:amount>
        <c4l:quantity>12.45</c4l:quantity>
        <c4l:uom>WHR</c4l:uom>
      </c4l:amount>
      <c4l:source>
http://semanco02.hs-albsig.de/repository/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Renewable\_Energy\_Source
      </c4l:source>
    </c4l:resourceElement>
```



```
</c4l:resourceList>
</ObjectEvent>
```

Disassembly Event

If a returned item can not be repaired or reused in any way, it is decomposed in order to recycle as much as possible of its constituents.

```
<TransformationEvent>
  <!--
    This is an example of a disassemble event that is used to record that an item is
    recycled to extract usable parts or raw materials.
  -->
  <eventTime>2019-05-27T10:00:10.124Z</eventTime>
  <eventTimeZoneOffset>+02:00</eventTimeZoneOffset>
  <inputEPCList>
    <epc>urn:epc:id:sgtin:4047111.012345.12345678902</epc>
  </inputEPCList>
  <outputQuantityList>
    <quantityElement>
      <epcClass>http://circ4life.eecc.info/material/copper</epcClass>
      <quantity>0.1</quantity>
      <uom>KGM</uom>
    </quantityElement>
  </outputQuantityList>
  <bizStep>urn:epcglobal:cbv:bizstep:destroying</bizStep>
  <disposition>urn:epcglobal:cbv:disp:sellable_not_accessible</disposition>
  <readPoint>
    <id>urn:epc:id:sgln:4047111.12345.0001</id>
  </readPoint>
  <bizLocation>
    <id>urn:epc:id:sgln:4047111.12345.0001</id>
  </bizLocation>
  <c4l:wasteList>
    <c4l:wasteElement>
      <c4l:waste>http://circ4life.eecc.info/material/plastics</c4l:waste>
      <c4l:amount>
        <c4l:quantity>0.5</c4l:quantity>
        <c4l:uom>KGM</c4l:uom>
      </c4l:amount>
      <c4l:sink>http://circ4life.eecc.info/sink/landfill</c4l:sink>
    </c4l:wasteElement>
  </c4l:wasteList>
</TransformationEvent>
```



Acknowledgment

EECC would like to thank all partners in CIRC4Life, in particular the collaborators in Work Package 4 and 5, for their valuable input and discussions. In particular thanks to ICCS for the productive collaboration in building the software architecture for the project as a whole are in order.

The structure of this development report uses elements from the arc 42 architecture framework, <http://www.arc42.de>, created by Dr. Peter Hruschka & Dr. Gernot Starke.

References

- [1] R. Tröger, S. Clanzett, R.J. Lehmann: *Innovative Solution Approach for Controlling Access to Visibility Data in Open Food Supply Chains*, DOI: <http://dx.doi.org/10.18461/pfsd.2018.1817>
- [2] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, P. Hallam-Baker: RFC 6920 *Naming Things with Hashes*, <https://www.rfc-editor.org/info/rfc6920>
- [3] GS1 Global, 2016. *EPC Information Services (EPCIS) Specification - Release 1.2*, Brussels: Global Standards One, <https://www.gs1.org/standards/epcis>
- [4] GS1 Global, *Core Business Vocabulary Standard - Release 1.2.2*, <https://www.gs1.org/standards/epcis>
- [5] GS1 Global, *EPC Tag Data Standard - Release 1.12*, <https://www.gs1.org/standards/epcrfid-epcis-id-keys/epc-rfid-tds/1-12>
- [6] EECC, *Open API Specifications of the CIRC4Life Traceability Module Endpoints*, <https://circ4life.eecc.info/doc/oas/>
- [7] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, E. Simon: *XML Signature Syntax and Processing - Version 1.1*, <https://www.w3.org/TR/xmlsig-core/>
- [8] J. Boyer: *Canonical XML - Version 1.0*, <https://www.w3.org/TR/2001/REC-xml-c14n-20010315>
- [9] T. Preston-Werner: *Semantic Versioning 2.0.0*, <https://semver.org/spec/v2.0.0.html>
- [10] EECC: *EPCIS Extension for Ecological Impacts - XSD*, <https://circ4life.eecc.info/epcis>



Appendix

1. M. Guenther, D. Woerner: *SupplyTree - A Federated Systems Approach to solve Supply Chain Traceability, Pre-Release for Circ4Life*, release version to be published on <https://www.bosch.com/research/know-how/success-stories/economy-of-things-a-technology-and-business-evolution/>
2. EECC: *Circ4Life Traceability Module - Open API Specification*, most recent version can be found at <https://circ4life.eecc.info/doc/oas/>

Appendix 1

SupplyTree - A Federated Systems Approach to solve Supply Chain Traceability

Pre-Release for Circ4Life

This is a pre-release for Circ4Life project. Released on 2019-07-25. More details about releasing department can be found at Robert Bosch GmbH, Economy of Things Project [3]

Authors

Matthias Guenther, Robert Bosch GmbH, Economy of Things

Dominic Woerner, Robert Bosch Schweiz, Economy of Things

Abstract / Executive Summary

In a global economy supply chains are complex networks with heterogenous participants. From the perspective of a finished product it is hard to trace down the individual suppliers of various hardware and software components. This is problematic in particular in case of a recall, where affected products and responsible suppliers have to be identified. There are different architectural patterns to approach this problem of supply chain traceability from centralized architectures to decentralized architectures based on blockchain technology and federated systems. Recently, blockchain-based systems have become popular due to their properties of auditability and immutability (i.e. tamper evidence). However, we will argue that these properties can also be achieved with a simpler, federated system called Supply Tree that utilizes concepts also common in blockchain systems, like cryptographic commitments, hash chains and digital signatures, but abandons the notion of a global shared state. In contrast to Blockchains which are heavily replicated and require lots of standardization or a common code base, Supply Tree is distributed and loosely-coupled and hence provides much better scaling behavior - both from a technological and from an organizational perspective.

Introduction

Supply chains are complex networks containing a multitude of participants around the world. Participants range from small and medium sized companies to multi-national cooperations. Today, most participants in the supply chain interact only with their immediate suppliers and customers, and data exchange

may happen over various channels from paper documents, e-mails and telephone calls to proprietary and standardized electronic interchange (EDI) solutions.

Due to the heterogenous nature of these networks it becomes very cumbersome, time consuming and costly to track down the origin of a malfunction that appears in a finished product.

Architectures for Supply Chain Traceability

In [1] a system for supply chain traceability is introduced as follows: “From an abstract viewpoint, a traceability information system can be thought of a single massive, centralized data storage capturing all the information about each lot along along each stage of the supply chain.”

However, the authors also point out that a centralized architecture has scaling issues and can hardly meet the dynamic requirements of a heterogenous group of supply chain participants. We may further add that a centralized architecture represents a single point of failure and thus a honeypot for attackers. In addition a central platform provided is a serious platform risk.

In recent years, decentralized architectures based on blockchain technology have become suggested as a novel alternative to the centralized architecture. Blockchains are peer-to-peer networks that are logically centralized but can be decentralized from a technical and organizational point of view. They are replicated state machines where state changes are represented by appending a set of signed transactions, a block. Blocks are chained by cryptographic hashes. Therefore, the history is auditable and tamper-evident. These properties of logical centralization, i.e. to have a single source of truth, and the auditability make the technology appealing for supply chain applications. However, in contrast to traditional distributed systems, workload and storage is not shared by the peers, but replicated. Hence, each participant carries the burden of all other participants. This overhead is paid for the property of having a global shared state which is not required in for supply chain traceability from our perspective.

Furthermore, in most supply chains, participants are not willing to share data with all participants and may only want to share because of specific contractual pressure. Therefore, additional measures have to be taken to shield sensitive data.

In this paper, we suggest a loosely-coupled decentralized system based on federation, where data is kept at the source and only links with cryptographic commitments are travelling downstream in the supply chain. Thereby data is only shared if required and the company that owns the data stays in control of the data. Still, the data becomes tamper-evident due to sharing of the commitment, and the supply chain becomes traceable because of linking. In contrast to centralized and blockchain architectures, the federated system is

much more scalable. This is obviously true from a technological point of view, since it is distributed in the traditional sense, but also from an organizational point of view, since the participants are only loosely coupled by a minimal set of standardized web APIs.

Supply Tree - the federated systems approach to supply chain traceability

Basic Concept

Supply tree is a combination of an append-only distributed data structure and a set of standardized APIs to create a federated system for the exchange of tamper-evident supply chain data. From the perspective of a finished product, a supply chain has the structure of a tree. The finished product represents the root node. The root node has directed edges to various child nodes representing different parts from different suppliers. This structure may continue until we end up with raw materials at the leaf nodes.

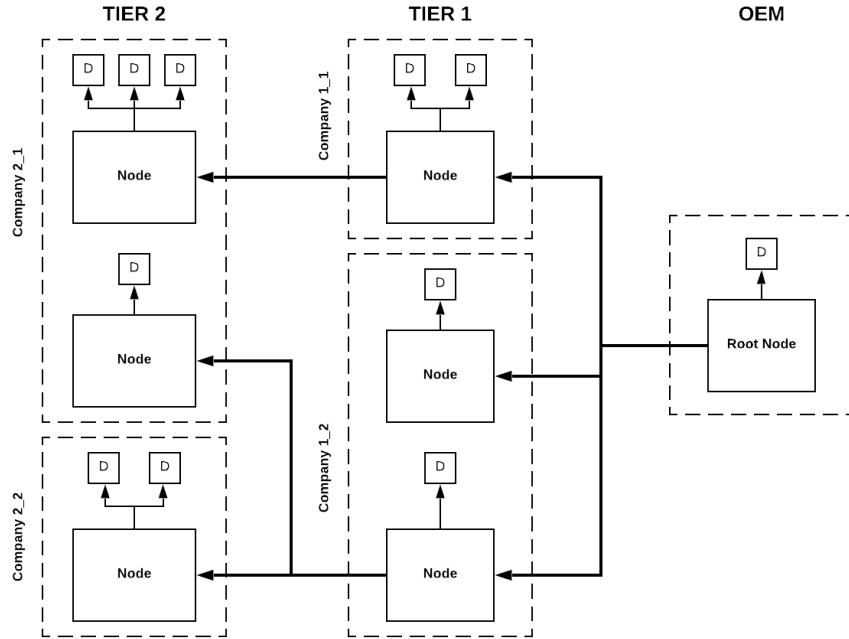


Figure 1: Supply Tree data structure

Each node is created by a company at the specific Tier X (Tier 2 in the picture) in the supply chain and is communicated to the next company in Tier X-1,

i.e. its customer (Tier 1 in the picture). Thereby the tree gets built up from the leafs to the root node over the production process. An edge is represented by a cryptographic hash of the respective child. Hence, the data structure becomes immutable, or more precisely, tamper-evident. A simple JSON representation of a node may look as follows

```
{
  "reference" : {
    "0" : "https://supplytree.tierX+1.com/tree/<SHA512_OF_TREE_OBJECT>",
    "1" : "https://supplytree.tierX+1.com/tree/<SHA512_OF_TREE_OBJECT>"
  },
  "data" : {
    "0" : "https://supplytree.tierX.com/data/<SHA512_OF_DATA_OBJECT>",
    "1" : "https://supplytree.tierX.com/data/<SHA512_OF_DATA_OBJECT>",
    "2" : "https://supplytree.tierX.com/data/<SHA512_OF_DATA_OBJECT>"
  },
  "uuid": "https://supplytree.tierX.com/uuid/9973494216984ee5f78a1d432105af1e5a3eb5c7",
}
```

The reference section contains the links to the child nodes. In addition, there is a data section. It contains a list of data objects that belong to the current node. These data objects are also referenced by cryptographic hashes in order to prevent modification of the data later on. Thus, a company down the supply chain has a commitment to the data but does not necessarily need to retrieve and store the data itself. Instead, the data can be retrieved if actually required, e.g. in the case of a recall. Then, the data object hash can be compared with the hash in the node object and any modification would become evident.

Although the node is uniquely identified by its cryptographic hash, there is the need for another unique identifier (uuid). This identifier allows to link the node to the physical item. At the point of production of the physical item an identifier gets imprinted. This identifier can't be the hash of the node object since the data belonging to the node might not be complete and adding data would lead to a new hash of the data object, and the node object subsequently.

Extensions

Non-repudiation with Digital Signatures

If we pick a specific point in the supply chain at Tier X. By sending the node object (or only its hash), the company in Tier X+1 commits to the data and is not able to change it without Tier X noticing. However, Tier X is able to modify the node object from Tier X+1 and could include the modified object in a node object that will be sent to Tier X-1. If these inconsistencies surface later on, it is hard to point out the responsible party. Therefore, node objects should be digitally signed and the receiving party should return a signed receipt.

Node Objects as capabilities for Access Control

In order to traverse a supply tree APIs of web servers hosted by several companies have to be accessed. Since the URLs of node and data objects include cryptographic hashes, the root node can be viewed as a capability to access to the resources along the tree. Some companies might be reluctant to rely solely on this kind of access control that allows everyone in the possession of the respective node object to access the data objects. In this case additional access control mechanisms can be implemented at the concerning API. However, we strongly suggest that the node resources stay unprotected, such that everyone down the supply chain is able to traverse the tree and can retrieve signed node objects. This entails that a company at a lower Tier gets information about the suppliers of its suppliers. Sometimes this might not be desired. Then access could also be restricted for the node resources and made only available to the direct customer or as an additional feature, only for request signed by a defined legal authority, e.g. Federal Motor Transport Authority (“KBA” in Germany).

Machine-readable Data Objects

Data objects in Supply Tree are binary blobs and the content is in principle unrestricted. In some cases this might be pdf or even images. In order to inform the client that accesses a specific data resource what to expect the content type should be defined. However, the real power can be unlocked if data objects are machine readable. Therefore, standardized JSON templates or JSON-LD documents are recommended.

IoT-based real-time Data

In some cases the transport of goods should be monitored in real-time or sensor data is collected over the course of the transport. These might be the location of the goods, concussions or environmental conditions, such as temperature or humidity. In this case a special type of data link can be added to the node object. This link can not be identified by its hash, because the content is changing. However, if the measurement values should be non-repudiable, then they should be digitally signed.

Example from the Automotive Supply Chain

In this example we’ll go through an automotive example to show how the flow of data would look like in a real world use case. We assume the following use case: An OEM receives an infotainment system to build it into the car. The infotainment system consists of several parts which are sourced from suppliers. In this case the ECU (Electronic Control Unit). This use case can be also seen as the **digital vehicle file** which documents the parts built into a specific car. The example consists of two parts. The first part is the production process

where the physical items travel along the supply chain and get assembled. At this stage detailed production information is typically not required upstream. The second part is a recall scenario, where a malfunction in a part downstream has been identified and the OEM has to identify which products are affected and which supplier is responsible.

Supply Chain Participants

- **Tier 2:** Delivers the ECU
- **Tier 1:** Assembles the ECU into the infotainment system and delivers to the OEM
- **OEM:** Assembles the infotainment system into the car.

Data flow

The production process

The data flow shows that Tier2 creates data objects, hashes them and includes those links into the node object. The data typically comes from existing data sources like a MES (Manufacturing Execution System) or ERP (Enterprise Resource Planning) system.

In the next step the node object is hashed and made available through a link. The link is sent to the next party in the supply chain, Tier1. This can be done in three ways: * either the node object itself is sent, * or the link to the node object is sent, * or the aforementioned UUID as a link is printed on the physical product through which the data is retrieved from the previous supply chain step.

In the example above, the link to the node object is sent to the next step in the chain.

In all cases, the messages should be signed and the receiver sends back a signed receipt.

Tier1 also creates its own data object and adds both, the link to the data object and the link to the Tier2 node object to a new node object. The new node object is hashed and sent as a link to the OEM.

The OEM does the same with its own data objects. The chain is now linked together and at any point in the future, any modifications to the objects can be detected and thus, be used as evidence in case of a dispute.

The recall

Now let's look into a recall situation. The OEM requests the node object content from Tier1. Tier1 sends back the content to the OEM. The OEM verifies, that the content matches the hash of the object to detect any possible

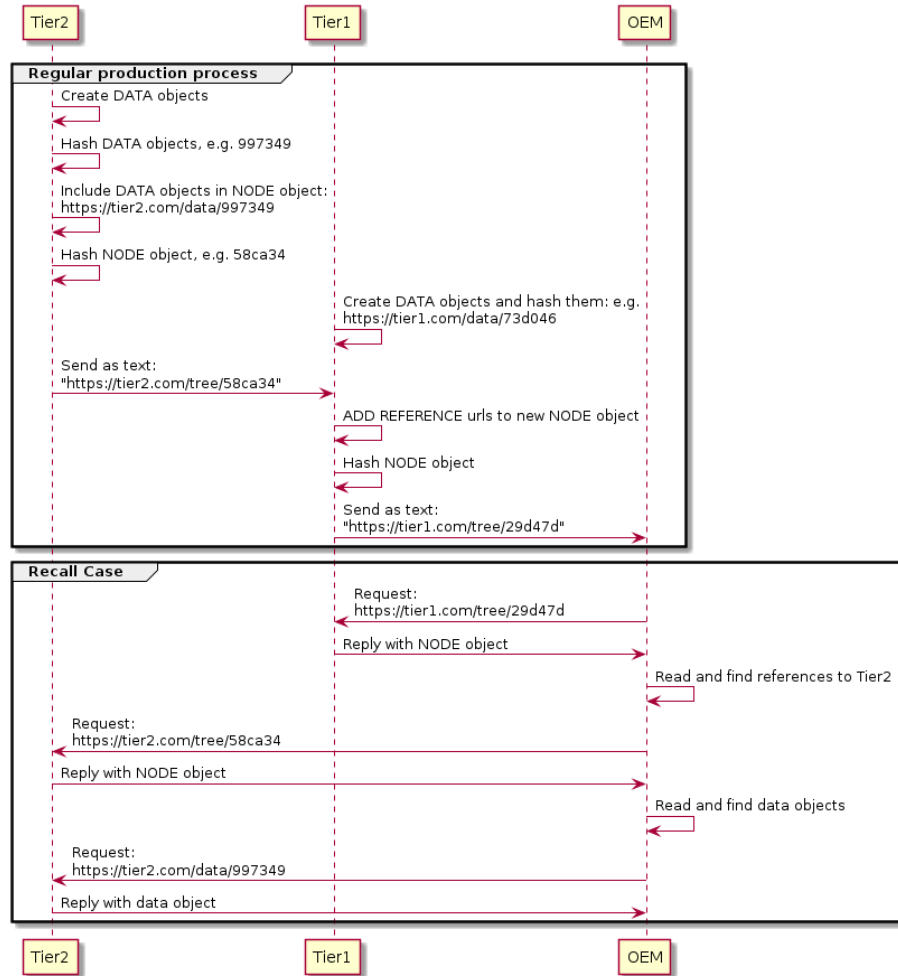


Figure 2: Automotive Supply Chain Example

modifications. Afterwards, the references to other node objects are parsed from the node object and a reference to the node object from Tier2 is detected. The OEM requests the Tier2 node object content, checks the hash and parses its content. The result shows a data object that is fetched from Tier2.

Because of the hashes and links between the objects, any change to the data since the time when the hashes were forwarded along the supply chain, can be detected.

In this example we assume, that the ECU was manufactured with material that later on has been identified being dangerous. Now, since the OEM can read this information, the OEM can recall only the affected cars from the market because of the given product instance data.

Relevant Objects

Node 58ca34

```
{
  "reference" : {
  },
  "data" : {
    "0" : "https://tier2.com/data/997349"
  }
}
```

Node 29d47d

```
{
  "reference" : {
    "0" : "https://tier2.com/tree/58ca34",
  },
  "data" : {
    "0" : "https://tier1.com/data/73d046"
  }
}
```

Applications

Integration into existing System Landscape

The picture shows how the integration of SupplyTree into an existing landscape would look like. On the left side, a simplified version of the **automation pyramid** [2] (page 5) shows the relevant IT systems in a factory. At the top, the EDI system shows how companies already communicate digitally in the **order**

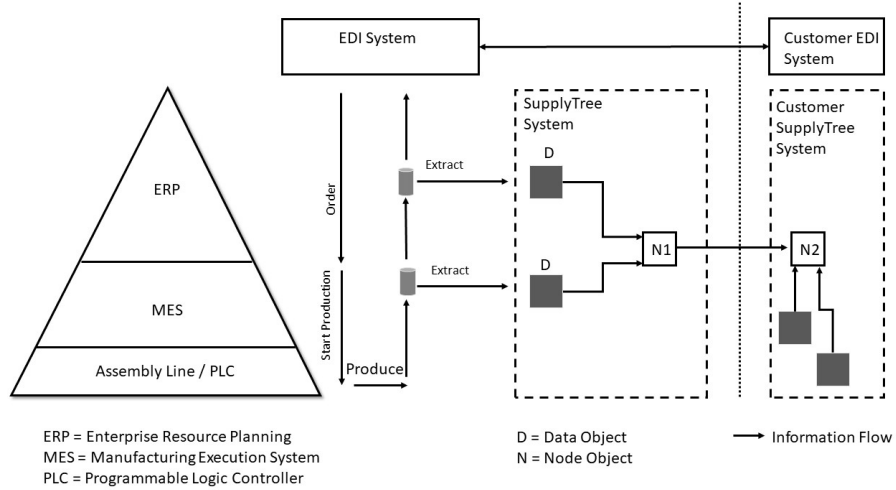


Figure 3: System Integration

to cash process. In the middle, the arrows show the information flow from an EDI system, through the different layers of the automation pyramid down to the assembly line and back up to the EDI system.

The SupplyTree System doesn't change this flow of information. It rather integrates well into this landscape. Data can be extracted from different existing systems, namely ERP and MES. The right side of the picture shows how SupplyTree uses the extracted data to create a tamper-evident traceable chain of that data, across multiple companies.

Implementation and Validation Status

We have implemented a first prototype allowing us to validate the concept. As a next step, we plan to build a production-ready, open source implementation of the concept and to test it in a real life supply chain use case.

Conclusion

SupplyTree explains a lightweight solution to solve trust issues along the supply chain. It does this by focusing on a tamper-evident data chain along the supply chain, that requires the participating parties to only agree on a minimalistic set of API standardization. No central - or logical central - instance and therefrom derived additional effort is required.



Circ4Life Traceability Module – Open API Specification

(C) Copyright 2018–2019 European EPC Competence Center GmbH (EECC)
<circ4life@eecc.info>

Version 2.2.0, 2019-07-26

Table of Contents

Overview	1
License information	1
URI scheme	1
Paths	1
Bin Disposal Event Receiving Endpoint	1
Bin Barcode Item Status Endpoint	2
Simplified Impact Capturing Endpoint	3
Object Event Capturing Endpoint	4
EPC Item Status Endpoint	5
Bin Barcode Total Impact Endpoint	6
EPC Item Total Impact Endpoint	6
SGTIN Total Impact Endpoint	7
Internal EPCIS Subscription Endpoint	8
Internal Inspection Event Endpoint	9
SGTIN Item Status Endpoint	10
Definitions	11
ApiResponse	11
BinDisposalData	11
BusinessTransaction	12
C4IObjectEPCEventData	12
Impact	13
ImpactData	13
InspectionEventUiData	14
Measure	15
QuantityElement	15
ResourceOrWaste	15
ResourceOrWasteList	16
SourceDestType	16
Status	17
Transportation	17

Overview

This is the open API specification for the Endpoints of Webservices developed and provided by the EECC within CIRC4Life. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776503-CIRC4Life-H2020-IND-CE-2016-2017/CIRC-2017/TwoStage.

License information

License : This document is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

License URL : <http://creativecommons.org/licenses/by-sa/4.0/>

Terms of service : THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, OR ANY WARRANTY OTHER WISE ARISING OUT OF THIS SPECIFICATION. EECC disclaims all liability for any damages arising from use or misuse of this Standard, whether special, indirect, consequential, or compensatory damages, and including liability for infringement of any intellectual property rights, relating to use of information in or reliance upon this document.

EECC retains the right to make changes to this document at any time, without notice. EECC makes no warranty for the use of this document and assumes no responsibility for any errors which may appear in the document, nor does it make a commitment to update the information contained herein.

URI scheme

Host : circ4life.eecc.info

BasePath : /

Paths

Bin Disposal Event Receiving Endpoint

```
POST /api/capture/bin-disposal-event
```

Description

Data from the intelligent bins control unit is send to this endpoint whenever an item is put into the bin.

Parameters

Type	Name	Description	Schema
Body	binDisposalData <i>required</i>	binDisposalData	BinDisposalData

Responses

HTTP Code	Description	Schema
200	Disposal event was successfully created and stored.	ApiResponse
400	The request is invalid. Likely there is a problem with the data/format. See message for details.	ApiResponse
500	Internal error: processing or storing the data failed. See message for details.	ApiResponse

Consumes

- [application/json](#)

Produces

- [application/json; charset=UTF-8](#)

Tags

- recycling-capture-controller

Bin Barcode Item Status Endpoint

```
GET /status/binbarcode/{binbarcode}
```

Description

The latest known location and disposition of an item can be queried here via its bin barcode.

Parameters

Type	Name	Description	Schema
Path	binbarcode <i>required</i>	binbarcode	string

Responses

HTTP Code	Description	Schema
200	Success	Status
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- [application/json; charset=UTF-8](#)

Tags

- status-controller

Simplified Impact Capturing Endpoint

POST /api/capture/impact/{epc}

Description

A simplified interface to enable the partners to record data about ecological impacts that is already accumulated and cannot be directly related to concrete events any more.

Parameters

Type	Name	Description	Schema
Path	epc <i>required</i>	epc	string
Body	impactData <i>required</i>	impactData	ImpactData

Responses

HTTP Code	Description	Schema
200	Impacts are recorded successfully.	ApiResponse

HTTP Code	Description	Schema
400	The request is invalid. Likely there is a problem with the data/format. See message for details.	ApiResponse
500	Internal error: processing or storing the data failed. See message for details.	ApiResponse

Consumes

- [application/json](#)

Produces

- [application/json; charset=UTF-8](#)

Tags

- impact-only-capture-controller

Object Event Capturing Endpoint

POST /api/capture/objectEvent

Description

This endpoint is capable of capturing any CIRC4Life Object event, but all parameters have to be specified explicitly.

Parameters

Type	Name	Description	Schema
Body	c4lObjectEvent Data <i>required</i>	c4lObjectEventData	C4lObjectEPCEvent Data

Responses

HTTP Code	Description	Schema
200	Event successfully captures.	No Content

HTTP Code	Description	Schema
400	Bad request, likely due to mal formatted input data. See message for details..	ApiResponse
500	Internal error in transforming/loading the event. See message for details.	ApiResponse

Consumes

- [application/json](#)

Produces

- [application/json; charset=UTF-8](#)

Tags

- event-capture-controller

EPC Item Status Endpoint

```
GET /status/epc/{epc}
```

Description

The latest known location and disposition of an item identified via its EPC.

Parameters

Type	Name	Description	Schema
Path	epc <i>required</i>	epc	string

Responses

HTTP Code	Description	Schema
200	Success	Status
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- `application/json; charset=UTF-8`

Tags

- `status-controller`

Bin Barcode Total Impact Endpoint

```
GET /impacts/binbarcode/{binbarcode}
```

Description

Returns the aggregated impacts for the given item identified by its bin barcode.

Parameters

Type	Name	Description	Schema
Path	binbarcode <i>required</i>	binbarcode	string

Responses

HTTP Code	Description	Schema
200	All known impacts associated with the given item are returned.	ImpactData
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- `application/json; charset=UTF-8`

Tags

- `impact-aggregation-controller`

EPC Item Total Impact Endpoint

```
GET /impacts/epc/{epc}
```

Description

Returns the aggregated impacts for the given item identified by its EPC.

Parameters

Type	Name	Description	Schema
Path	epc <i>required</i>	epc	string

Responses

HTTP Code	Description	Schema
200	All known impacts associated with the given item are returned.	ImpactData
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- [application/json; charset=UTF-8](#)

Tags

- impact-aggregation-controller

SGTIN Total Impact Endpoint

```
GET /impacts/sgtin/{gtin}/{serial}
```

Description

Returns the aggregated impacts for the given item identified by its SGTIN.

Parameters

Type	Name	Description	Schema
Path	gtin <i>required</i>	gtin	string

Type	Name	Description	Schema
Path	serial <i>required</i>	serial	string

Responses

HTTP Code	Description	Schema
200	All known impacts associated with the given item are returned.	ImpactData
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- [application/json; charset=UTF-8](#)

Tags

- impact-aggregation-controller

Internal EPCIS Subscription Endpoint

POST /internal/subscription

Description

Receives EPCIS standard (i.e. XML) event lists from subscriptions.

Parameters

Type	Name	Description	Schema
Body	newEvents <i>required</i>	newEvents	string

Responses

HTTP Code	Description	Schema
200	All events successfully processed	string

HTTP Code	Description	Schema
500	Error processing some events. Please re-send next time.	string

Consumes

- [application/xml](#)
- [text/xml](#)

Produces

- [application/xml](#)
- [text/xml](#)

Tags

- subscription-accepting-controller

Internal Inspection Event Endpoint

POST /api/capture/inspection-event

Description

Used to receive data from the recycling inspection web user interface.

Parameters

Type	Name	Description	Schema
Body	inspectionEventUiData <i>required</i>	inspectionEventUiData	InspectionEventUiData

Responses

HTTP Code	Description	Schema
200	Inspection event was successfully created	ApiResponse
400	The request is invalid. Likely there is a problem with the data/format. See message for details.	ApiResponse

HTTP Code	Description	Schema
500	Internal error: processing or storing the data failed. See message for details.	ApiResponse

Consumes

- [application/json](#)

Produces

- [application/json; charset=UTF-8](#)

Tags

- recycling-capture-controller

SGTIN Item Status Endpoint

```
GET /status/sgtin/{gtin}/{serial}
```

Description

The latest known location and disposition of an item can be queried here via its SGTIN.

Parameters

Type	Name	Description	Schema
Path	gtin <i>required</i>	gtin	string
Path	serial <i>required</i>	serial	string

Responses

HTTP Code	Description	Schema
200	Success	Status
400	Bad request. (No ID or mal formatted ID given.)	ApiResponse
404	Item not found.	ApiResponse

Produces

- `application/json; charset=UTF-8`

Tags

- `status-controller`

Definitions

ApiResponse

Generic response including a message for debugging errors.

Name	Description	Schema
message <i>optional</i>	Specific error message. Required if the status is not 2xx. Example : <code>"Write-only-memory subsystem too slow for this machine."</code>	string
path <i>required</i>	The requested path. Example : <code>"/"</code>	string
reason <i>optional</i>	Standard human readable reason for the HTTP status code. Example : <code>"Ok"</code>	string
status <i>optional</i>	Same code as returned in the HTTP header. Repeated here for logging/debugging purposes. Example : <code>200</code>	integer (int32)
timestamp <i>optional</i>		string (date-time)

BinDisposalData

This format is used to send data about disposal of an item from the intelligent bin to the Traceability Module.

Name	Description	Schema
binBarcode <i>required</i>	The same number as encoded into the printed bar code label. Contains the RecycleBinUserID as the first 11 digits. Example : <code>"1600400000118112200011"</code>	string

Name	Description	Schema
eventTime <i>optional</i>	Time when the event happens Example : "2019-05-16T13:40:00+02:00"	string (date-time)
fillingLevel <i>optional</i>	Filling level of the bin at the time of disposal in % Example : 42.23	number (double)

BusinessTransaction

According to the EPCIS standard.

Name	Description	Schema
bizTransaction <i>required</i>	Standard BusinessTransactionID Example : "http://transaction.acme.com/shipment/34ABC8"	string
type <i>optional</i>	Standard BusinessTransactionTypeID	string

C4LObjectEPCEventData

This is a JSON representation of an EPCIS object event as used in CIRC4Life. The Eco Impact Extension is explicitly included.

Name	Description	Schema
action <i>required</i>	See EPCIS standard for details. Example : "ADD"	enum (ADD, OBSERVE, DELETE)
bizLocation <i>optional</i>	Standard epcis:BusinessLocationType	string
bizStep <i>optional</i>	Standard epcis:BusinessStepIDType	string
bizTransaction List <i>optional</i>	Standard epcis:BusinessTransactionListType	< BusinessTransaction > array
destinationList <i>optional</i>	Standard epcis:DestinationListType	< SourceDestType > array
disposition <i>optional</i>	Standard epcis:DispositionIDType	string

Name	Description	Schema
epcList <i>optional</i>	Standard epcglobal:EPC list (epcs are to be represented in URI format) Example : <code>"urn:epc:id:sgtin:4047111.012345.012345678901"]</code>	< string > array
eventTime <i>required</i>	When the event happened. (Contains offset)	string (date-time)
ilmd <i>optional</i>	epcis:ILMDType	< object > array
impactData <i>optional</i>	Data about the ecological or other impacts associated with the event	ImpactData
quantityList <i>optional</i>	Standard epcis:QuantityListType	< QuantityElement > array
readPoint <i>optional</i>	Standard epcis:ReadPointType	string
sourceList <i>optional</i>	Standard epcis:SourceListType	< SourceDestType > array

Impact

A concrete ecological (or other) impact. This is the JSON representation of <c4l:impactElementType>.

Name	Description	Schema
impactType <i>required</i>	This is a URI as specified in <c4l:impactType>. Example : <code>"http://dbpedia.org/resource/Premises"</code>	string
measure <i>optional</i>	Quantifying the amount	Measure

ImpactData

Data about product specific ecological impacts. Contains any elements from the c4l epcis extension.

Name	Description	Schema
impactList <i>optional</i>	JSON representation of <c4l:impactList>. It can be used to track social or ecological impacts that are not associated with material flows such as e.g. land usage.	< Impact > array
resourceList <i>optional</i>	This list represents the inflow of material into the process. It is the JSON representation of <c4l:resourceList>.	ResourceOrWasteList
transportationList <i>optional</i>	This is the JSON representation of <c4l:transportList>. If the environmental impact of transportation can not be broken down into resource consumption (fuel,...) and type (exhaust fumes,...) then a transportList can be used instead.	< Transportation > array
wasteList <i>optional</i>	This represents the outflow (typically not including the actual products) of a process. JSON representation of <c4l:wasteList>.	ResourceOrWasteList

InspectionEventUiData

Data Received from the UI in order to create an inspection event

Name	Description	Schema
age <i>optional</i>	Estimated lifetime in ISO duration format. Need to be set if state is REUSABLE, BROKEN or WORKING Example : " P10Y "	string
binBarcode <i>required</i>	Barcode from the bin label. Must be at least 22 digits long Example : " 16004000001181122000011 "	string
brand <i>optional</i>	Brand of the item. Need only to be filled out if GTIN is missing and product is a EEE item Example : " Apple "	string
gtin <i>optional</i>	GTIN of the item. Required if product is a EEE item. If GTIN is not available, please provide manufacturer & model Example : " 4047111000006 "	string
model <i>optional</i>	Model of the item. Need only to be filled out if GTIN is missing and product is a EEE item Example : " iPhone 7 "	string

Name	Description	Schema
serialNumber <i>optional</i>	Serialnumber of the item. Required if product is a EEE item Example : "X01X23Y4XYXY"	string
state <i>required</i>	Condition of the electronic device Example : "WORKING"	enum (WORKING, REUSABLE, BROKEN, NO_EEE_ITEM)
timestamp <i>required</i>	Timestamp of the inspection Example : "2019-05-16T13:40:00+02:00"	string (date-time)
unuKey <i>optional</i>	UNUkey of the electronic device. Need to be set if state is REUSABLE, BROKEN or WORKING Example : "303"	enum (303, 306, 401, 402, 406, 701, 702)

Measure

Data about product specific ecological impacts

Name	Description	Schema
quantity <i>required</i>	Amount, measured in UOM. Example : 1294.5	number
uom <i>optional</i>	Unit of Measure. JSON representation of <c4l:measure>. May be omitted if there is a natural uom. Example : "MTK"	string

QuantityElement

JSON form of the EPCIS standard epcis:QuantityElementType

Name	Description	Schema
epcClass <i>required</i>	URI of the epc class, see standard.	string
measure <i>optional</i>	Contains uom and quantity as in the epcis standard, although not named measure there.	Measure

ResourceOrWaste

Information about a specific type of waste produced or a resource consumed. Implements

c4l:wasteElementType and c4l:wasteElementType.

Name	Description	Schema
amount <i>required</i>	c4l:measure amount of waste produced/resource consumed	Measure
sourceOrSink <i>optional</i>	c4l:sinkType or c4l:sourceType: Semantic web URI for how the waste was deposited/from where the resource is taken. Example : "http://semanco02.hs-albsig.de/repository/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Not-Renewable_Energy_Source"	string
type <i>required</i>	c4l:wasteType or c4l:resourceType: URI for the type of produced waste/consumed resource. Preferably using a semantic web URI. Example : "https://w3id.org/saref#Electricity"	string

ResourceOrWasteList

This type represents a material flow and can be used to implement c4l:resourceList and c4l:wasteList. Notice that, as opposed to the XML Version, this JSON/Java version has 3 Lists of elements of one type instead of one list containing 3 different types.

Name	Description	Schema
epcList <i>optional</i>	Standard epcglobal:EPC list (epcs are to be represented in URI format) Example : "urn:epc:id:sgtin:4047111.012345.012345678901"]	< string > array
quantityList <i>optional</i>	Standard epcis:QuantityElementType list.	< QuantityElement > array
resourceOrWasteList <i>optional</i>	CIRC4Life specific type element format	< ResourceOrWaste > array

SourceDestType

According to the EPCIS standard.

Name	Description	Schema
sourceDest <i>required</i>	Standard SourceDestID. An identifier that denotes a specific source or destination.	string

Name	Description	Schema
type <i>optional</i>	Standard SourceDestTypeID. An identifier that indicates what kind of source or destination this Source or Destination (respectively) denotes.	string

Status

The latest known status of an item

Name	Description	Schema
disposition <i>required</i>	Last known disposition (status) of the item Example : "urn:epcglobal:cbv:disp:returned"	string
epc <i>required</i>	The items identifier in EPC (URN) Format Example : "https://circ4life.eecc.info/recycling/bin/obj/1600400000118112200001100001"	string
location <i>required</i>	Last known location of the item Example : "urn:epc:epc:sgln:4057847.0000018.0"	string

Transportation

JSON Format for c4l:transportElementType.

Name	Description	Schema
distance <i>optional</i>	Distance travelled.	Measure
vehicle <i>required</i>	c4l:vehicleType URI naming the vehicle type used for transportation Example : "http://purl.org/vso/ns#Truck"	string

References

- [1] Bechini, Alessio, et al. “Patterns and technologies for enabling supply chain traceability through collaborative e-business.” *Information and Software Technology* 50.4 (2008): 342-359.
- [2] Hollender, Martin. Collaborative process automation systems. ISA, 2010.
- [3] <https://www.bosch.com/research/know-how/success-stories/economy-of-things-a-technology-and-business-evolution/>